Diplomarbeit

Durchgängige Modellierung: Vorgehen und Funktionalität im GPM-Tool

Steve Rechtenbach

1. Gutachter: Prof. Dr. Manfred Reichert
2. Gutachter: Dr. Thomas Bauer, Daimler AG
Betreuer: Stephan Buchwald, Daimler AG
Durchgeführt bei: Daimler AG

Vorwort

Diese Arbeit entstand im Jahr 2010 als Diplomarbeit bei der Daimler AG.

Mein besonderer Dank gilt Herrn Stephan Buchwald, der mich im Zuge meiner Werkstudententätigkeit bei der Daimler AG für dieses Thema begeistern konnte, und mich noch weit über seine Tätigkeit als Ansprechperson hinaus unterstützt hat. Des Weiten möchte ich mich speziell bei Herrn Prof. Dr. Manfred Reichert sowie Herrn Dr. Thomas Bauer für die Unterstützung und Übernahme des Erst- und Zweitgutachtens bedanken.

Insbesondere gilt mein Dank meinem Vater Michael Rechtenbach und seiner Frau Petra Döbert für den moralischen Beistand während meiner gesamten Studienzeit.

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sinngemäße Übernahmen aus anderen Werken sind als solche kenntlich gemacht und mit genauer Quellenangabe (auch aus elektronischen Medien) versehen.

Ulm, den 14. Juni 2010

Steve Rechtenbach

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Im Zusammenhang mit Business-IT-Alignment ist es wichtig die durchgehende Modellierung von Geschäftsprozessen auf der fachlichen Ebene und deren Abbildung auf die bestehende bzw. geplante technische Infrastruktur(IT-Ebene) zu verbessern. Aus fachlicher Perspektive steht die möglichst ausführliche Beschreibung der beteiligten Teilprozesse im Vordergrund. Im fachlichen Prozessmodell werden daher die Funktionalitäten der einzelnen Dienste in den Mittelpunkt gestellt, und die technischen Details vorerst ausgeblendet. In Hinblick auf die Ausführbarkeit des Prozesses (beispielsweise durch eine Workflow Engine) muss jedoch der Fachprozess in ein Metamodell überführt werden, welches die ausführungsrelevanten Informationen enthält.

Diese semantische Hürde zwischen Fachaktivitäten-Beschreibung und
Systemaktivitäten-Funktionalität muss durch Abstimmungen zwischen Fach- und IT-Bereich überwunden werden. Um den Aufwand für die Abstimmung mit Hilfe einer formalen Beschreibung zu verringern, muss eine ausführliche Dokumentation der Überführung unter den Modellen erfolgen, da der Fachprozess auf der technischen Ebene oft stark umstrukturiert wird.
Aus diesem Grund wird zwischen dem fachlichen Prozessmodell und dem ausführbaren Modell das Systemmodell eingefügt. Dieses besteht aus dem Systemprozess und dem Abbildungsmodell. Ersteres stellt eine implementierungsunabhängige Sicht auf den ausführbaren Prozess dar. Das Abbildungsmodell enthält unabhängig davon die Operationen (Transformationen) um die Umstrukturierung des Fachprozesses auf den Systemprozess zu dokumentieren.

1.1 Zielsetzung der Arbeit

Ziel dieser Arbeit ist, die bidirektionale Nachvollziehbarkeit der Transformationen zwischen Fachmodell-Aktivitäten und Systemmodell-Aktivitäten mit Hilfe des Abbildungsmodells sicherzustellen. Dadurch werden Änderungen transparent und können schnell identifiziert werden.

Es wird untersucht, wie das Abbildungsmodell grundlegend erstellt werden kann, und welche Pro-

bleme dabei auftreten. In diesem Zusammenhang werden verschiedene Methodiken betrachtet, wie sich das Abbildungsmodell aus Fach- und Systemmodell ableiten lässt, bzw. die Modellierung des Systemmodells gar mit Hilfe des Abbildungsmodells unterstützt werden kann.

Abschließend sollen die Erkenntnisse zur Gestaltung des Abbildungsmodells in einem Prototyp umgesetzt werden. Hierbei wird eine der zuvor untersuchten Methoden ausgewählt, und anhand eines Anwendungsszenarios durchgängig modelliert.

1.2 Anwendungsszenario

An dieser Stelle wird zunächst ein abstraktes Anwendungsszenario eingeführt, welches die Notwendigkeit für Durchgängigkeit in der Prozessmodellierung illustrieren soll. Auf dieses Beispiel wird in den folgenden Kapiteln des Öfteren eingegangen, weshalb an dieser Stelle die einzelnen Teilschritte ausführlich beschrieben werden.

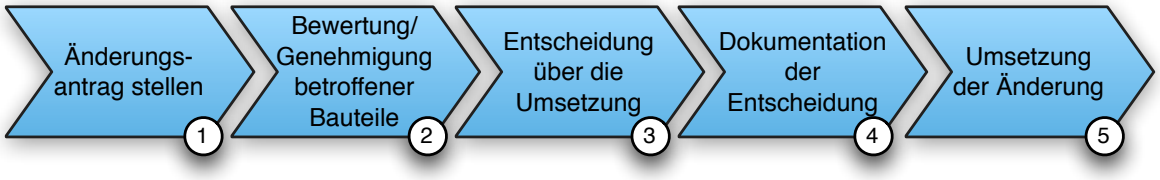

Abbildung 1.1: Beipielprozess als Wertschöpfungskette

Der Prozess beschreibt den Vorgang für die Änderung eines oder mehrerer Einzelteile einer Fahrzeugvariante, wie sie in ähnlicher Form bei Daimler verwendet wird [BBR09]. Fahrzeuge werden normalerweise in einer Grundkonfiguration angeboten, besitzen jedoch eine gewisse Variabilität in der Ausstattung. Beispielsweise kann ein Fahrzeug der gleichen Baureihe mit Diesel- oder Benzinmotor ausgeliefert werden. Von der Änderung des Motortypen sind einzelne Teile wie zum Beispiel die Abgasanlage betroffen, weshalb bei so einer Konfigurationsänderung eine Abstimmung mit Technikern erforderlich ist.
In dem oben mittels einer Wertschöpfungskette dargestellten Beispielprozess wird zumächst formal ein Änderungsantrag gestellt ① der in einem nächsten Schritt vom Technikbereich genehmigt werden muss ②. Ist dies geschehen werden direkt oder indirekt betroffene Bauteile identifiziert und die technische Umsetzung geplant ③. In einem Variantenmanagement-Programm wird die Konfiguration erstellt und die abhängigen Fahrzeugteile dokumentiert ④.
In einem letzten Schritt kann nun auf Basis der Planung die fertige Konfiguration an die Fertigung übergeben werden ⑤.

1.3 Aufbau der Arbeit

Im folgenden Kapitel 2 werden verschiedene Begriffe behandelt, die dafür verwendet werden die Notwendigkeit durchgängiger Modellierung darzustellen. Anhand dieser werden die Rahmenbedingungen für diese Arbeit festgelegt und verwendete Notationen erklärt. Wie bereits beschrieben, wird der Ansatz des Systemmodells vorgestellt um fachliche und technische Sicht eines Geschäftsprozesses einander näher zu bringen. In diesem Zusammenhang werden die Bestandteile des Systemmodells, wie das Abbildungsmodell und der Systemprozess, ausführlich behandelt.

Unabhängig von dem verwendeten Prozessmodellierungswerkzeug werden in Kapitel 3 vier Methoden vorgestellt, wie das Abbildungsmodell erstellt bzw. generiert werden kann. Die ersten beiden Methoden befassen sich mit der allgemeinen Herangehensweise für die Erstellung des Abbildungsmodells, wobei als Ausgangspunkt aufgrund des Business-IT-Alignments jeweils der Fachprozess dient. In den beiden letzten Methoden werden Automatismen untersucht, die die Erstellung des Abbildungsmodells vereinfachen können. Diese haben zum Ziel, dass einerseits der Aufwand für die Erstellung des Abbildungsmodells verringert werden kann, und andererseits die Modellierungsfehler bzw. Inkonsistenzen zwischen Fach- und Systemprozess schon während der Erstellungsphase erkannt werden können.

In Kapitel 4 wird nach der initialen Erstellung das Abbildungsmodell dafür verwendet, um bei unvorhergesehenen Änderungen in Fach- oder Systemprozess den zuständigen Bereich der jeweils andere Prozessebene zu benachrichtigen. Dies geschieht dadurch, dass spontane Änderungen in den Prozessen durch das Abbildungsmodell als Inkonsistenz erkannt werden, wodurch agil sowie flexibel in der betroffenen Modellebene darauf reagiert werden kann.

Abschließend wird in Kapitel 5 anhand eines Prototypen eine ausgewählte Methodik für die Erstellung des Abbildungsmodells in einem der untersuchten Werkzeuge umgesetzt. Dadurch soll die Unabhängigkeit des Abbildungsmodells von verschiedenen Modellierungssprachen, sowie die Umsetzbarkeit in zurzeit verwendeten Werkzeugen demonstriert werden.

2 Begriffe

An dieser Stelle werden verschiedene Begriffe behandelt, die zum Verständnis dieser Arbeit notwendig sind. Dazu zählen allgemeine Beschreibungen des Sachverhalts sowie spezielle Notationen auf die im späteren Verlauf des öfteren zurückgegriffen wird. Im Anschluss daran werden die Rahmenbedingungen für die Arbeit erarbeitet und das bereits eingeführte abstrakte Anwendungsszenario von der fachlichen bis zu technischen Ebene vorgestellt.

2.1 Geschäftsprozesse

Geschäftsprozesse beschreiben einen unternehmensbezogenen Arbeitsablauf, der mehrmals wiederholt werden kann. Sie bestehen aus einzelnen Aktivitäten, die einen elementaren Arbeitsschritt, oder ganze Arbeitsabläufe beschreiben. Zusammengefasst in einem Geschäftsprozess sind sie Teil der Ablauforganisation eines Betriebes und somit ein wichtiger Bestandteil bei der Erzeugung eines Produktes, oder einer Leistung. Geschäftsprozesse beschreiben konkret, *was* passieren muss um zu dem Ergebnis, also dem fertigen Produkt, zu kommen.

Der Schwerpunkt liegt auf der Darstellung des Ablaufs sowie der Koordination der einzelnen fachlichen Aktivitäten. Diese Prozesse werden daher optimiert, um einerseits wirtschaftlich effizient zu sein und Kosten zu sparen, des weiteren möglichst wenig Ressourcen zu verbrauchen, und andererseits so schnell wie möglich durchlaufen zu werden.

2.2 Geschäftsprozessmodellierung

Um diese Optimierung durchführen zu können muss der Geschäftsprozess formalisiert werden. Dies geschieht mit Hilfe der Geschäftsprozessmodellierung[BMW09]. Der Ablauf wird meist graphisch beschrieben, um ein Gesamtbild auf alle betroffenen Bereiche zu erhalten und auf deren Basis mögliche Stellen für die Optimierung zu identifizieren.

In der Geschäftsprozessmodellierung wird für jeden Geschäftsprozess einen Prozess-Lebenszyklus definiert, der durchlaufen wird (Abbildung 2.1).

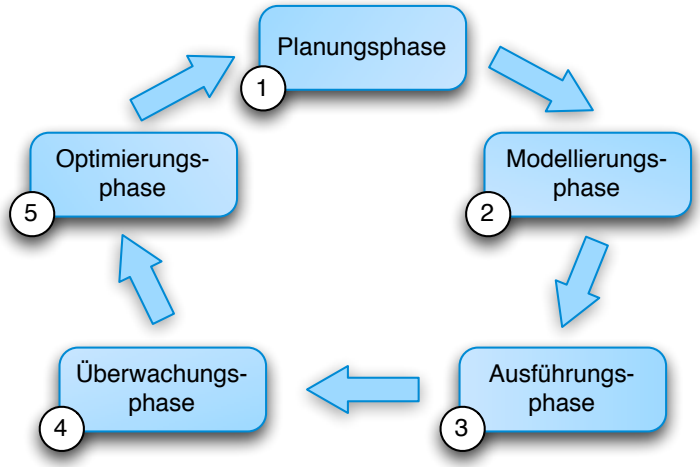

Abbildung 2.1: Process Lifecycle

Dieser Lebenszyklus besteht aus mehreren Phasen der sich von der fachlichen bis zu technischen Ebene erstrecken, und auf die im folgenden eingegangen werden soll [Wes07].

① **Planungs- und Analysephase:** In dieser Phase werden die einzelnen Aktivitäten eines Geschäftsprozesses identifiziert. Diese können entweder bestehende Aktivitäten sein, oder im Zuge der Neuplanung eines Geschäftsprozesses von Grund auf neu entworfen werden. Bereiche die dabei in den Planungsprozess einbezogen werden, sind bspw. Bearbeiterzuordnungen, Service-Level-Agreements (SLA's) und Vorüberlegungen in welcher Form der Prozess repräsentiert werden soll.
Bei der Neuplanung können sich je nach Vorgehen bei der Geschäftsprozessmodellierung die fachlichen Prozesse aus der bestehenden IT-Infrastruktur ableiten, oder sich die technischen Prozesse nach den Vorgaben der fachlichen Modellierung richten („Bottom-Up“ vs. „Top-Down“[ZZ09]).

② **Modellierungsphase:** In der Modellierungsphase wird der geplante Prozess konkret in einer Modellierungsnotation beschrieben. Der so entstandene Prozess wird im Anschluss daran syntaktisch und semantisch validiert, um einerseits Fehler in der Modellierung zu finden, und um andererseits zu überprüfen, ob der neu entstandene Prozess alle Planungsziele unterstützt. Ein Hilfsmittel um semantische Probleme zu finden ist die Simulation des Prozesses. Hierbei werden verschiedene Szenarien durchgelaufen, um im Prozess Stellen zu finden, die nicht das gewünschte Verhalten aufweisen.

③ **Ausführungsphase:** Der formalisierte Prozess wird nun in Form von Instanzen ausgeführt. Typischerweise wird bei einem Startereignis eine neue Instanz des definierten Workflow's gestartet, die von dem für die Ausführung verantwortlichen Workflow-Management-System verwaltet

werden [vdAvH04]. Dieses System ist außerdem für die korrekte Ausführung sowie Reihenfolge der einzelnen Aktivitäten verantwortlich.

④ **Überwachungsphase:** Die Überwachungsphase visualisiert und überwacht einerseits den Status der einzelnen Prozessinstanzen, und sammelt andererseits Informationen über die einzelnen Prozessschritte. Die dabei gewonnenen Informationen legen die Basis für die Feststellung der Auslastung der beteiligten Stationen (Service, Bearbeiter) und bei der späteren Optimierung des laufenden Prozesses.

⑤ **Optimierungsphase:** Die in der vorigen Phase durch bspw. Process-Mining [vdAvDH+03] gewonnenen Daten werden ausgewertet und fließen in die nächste Verbesserungsphase des Prozesses ein. Die hier gewonnenen Laufzeitdaten bilden somit die Grundlage für die nächste Planungsphase.

2.3 Service-orientierte Architektur - SOA

Eine SOA [BBP09, Erl07] ist eine Art Denkmuster, oder Sammlung von Architekturprinzipien für die Realisierung und Pflege von Geschäftsprozessen durch verteilte Systeme [Jos08].
Die frühere Herangehensweise für die Unterstützung eines Unternehmens durch die IT war es, in einem Geschäftsprozess einen klar definierten Abschnitt zu identifizieren, welcher durch ein System oder eine Anwendung umgesetzt wird. Dies war eine leicht kalkulierbare Anschaffung mit einem klar definierten Geschäftswert bzw. Mehrwert.
Bei Änderungen in der Geschäftsprozessstruktur ist es jedoch schwer ein starres System kostengünstig zu ändern [Wes07, Rei00, RD00]. Je nach Umfang der Umstrukturierung des Geschäftsprozesses ist es häufig günstiger die Zusatzfunktionalität durch ein neues System anzubieten. Dadurch sammeln sich jedoch mit der Zeit immer mehr Anwendungen an, die verschiedenste Funktionen und Aufgaben ausführen. Die Probleme sind längerfristig gesehen gravierend.

Bei voneinander unabhängigen Systemen endet dies in der redundanten Implementierung von bereits existierender IT-Lösungen. Bei voneinander abhängigen Systemen kommunizieren diese über die unterschiedlichsten Schnittstellen, was zu einer schweren Änderbarkeit dieser untereinander hart verdrahtetem Anwendungen führt. Das resultiert im Endeffekt in einem enormen Wartungsaufwand vieler heterogener Systeme und der langsamen Reaktion auf Änderungen von Geschäftsprozessen.

Eine SOA soll dem Unternehmen die Möglichkeit bieten flexibel einen Geschäftsprozess zu ändern, und den IT-Bereich effizient daran anzupassen.
Dies geschieht dadurch, dass verteilte Anwendungen durch Middleware-Techniken in Dienste (Services) gekapselt werden. Services sind physisch unabhängige Programme, die nach außen hin eine Funktionalität anbieten, jedoch von ihrer Implementierung abstrahieren [Erl07]. Zusammenfassend führt die Verwendung von Services in einer SOA dabei zu folgenden Vorteilen:

- Verringerte Abhängigkeit der Anwendungssysteme untereinander (lose Kopplung von Services)
- Abstraktion von Implementierungsdetails der Services

- Wiederverwendbarkeit von Services
- Einsatz der Services in unterschiedlichen Konfigurationen
- Erhöhte Vorhersagbarkeit des Verhaltens von Services im Vergleich zu den bisherigen Anwendungen
- Erhöhte Erreichbarkeit und Skalierbarkeit der IT
- Leichtere Erkennbarkeit der Funktionen der Services durch Vereinbarungen über Schnittstellen zwischen Servicenutzer und Serviceanbieter

2.4 Notationen

Im Zuge dieser Arbeit werden verschiedene Notationen für die Repräsentation der Prozesse auf den unterschiedlichen Ebenen des verwendet. Zu diesen zählen die „Ereignisgesteuerten Prozessketten“, die „Business Process Modeling Notation“, die „Unified Modeling Language“ und die „Business Process Execution Language“. Daher werden diese nachfolgend einzeln vorgestellt.

2.4.1 Ereignisgesteuerte Prozessketten:

Die Ereignisgesteuerte Prozesskette (EPK) ist eine semiformale Modellierungssprache, die 1992 aus einem Forschungsprojekt unter der Leitung von August-Wilhelm Scheer hervor gegangen ist[Sch01]. EPK's werden heutzutage von vielen verschiedenen Werkzeugen unterstützt, und sind vor allem im deutschsprachigem Raum sehr verbreitet. Unter anderem basieren sie auf den Petri-Netzen [vdA98], wurden jedoch um neue Symbole und Semantik erweitert, um die fachlichen Abläufe bzw. den Geschäftsprozess in Unternehmen besser darzustellen.

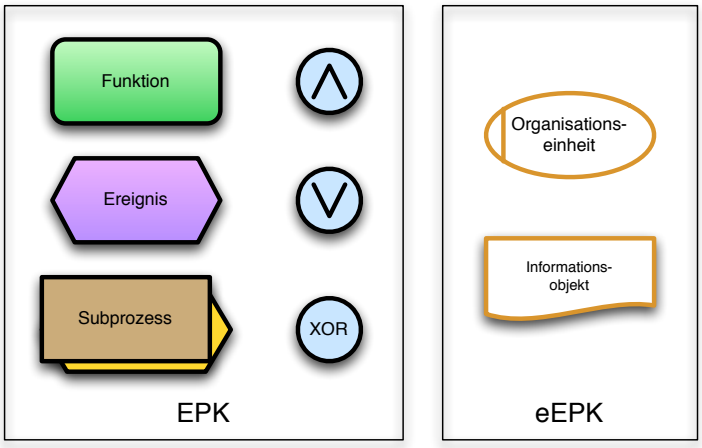

Abbildung 2.2: Ereignisgesteuerte Prozessketten

Um einen Geschäftsprozess zu modellieren werden Ereignisse und Funktionen verwendet. Einer Funktion geht immer ein Ereignis voraus, welches diese einleitet, und einer Funktion folgt im-

mer ein Ereignis. Ein Ereignis ist ein Zustand innerhalb des Geschäftsprozesses, wohingegen eine Funktion die eigentliche Einzelaktivität beschreibt, um diesen Zustand zu erreichen. Der Kontrollfluss einer EPK, der Reihenfolge der Funktionen und Ereignisse festlegt, kann durch sog. Konnektoren aufgespalten, bzw. zusammengefasst werden. Hierfür sind 3 verschiedene Typen (AND, OR, XOR) definiert (vgl. Abbildung 2.2).

Die erweiterten Ereignisgesteuerten Prozessketten (eEPK) wurden durch Elemente wie Informationsobjekte (Datenobjekte) und Organisationseinheiten erweitert, wodurch komplette unternehmensrelevanten Aspekte[1] modelliert werden können.

2.4.2 Unified Modeling Language - UML

Die Unified Modeling Language nimmt sich zum Vorsatz, die Verbindung zwischen fachlichen und technischen Anforderungen mit einer Vielzahl von Modellen durchgehend herzustellen [OMG05]. Ursprünglich wurde UML für die formale Modellierung von Softwaresystemen entwickelt, um diese über alle Konzeptionsebenen einheitlich beschreiben zu können. UML befindet sich zum Stand dieser Diplomarbeit in der Version 2.2, wobei nach aktuellem Zeitplan der OMG[2] Version 2.3 dieses Jahr erscheinen soll.

UML benutzt eine Reihe von Diagrammtypen um Elemente der Geschäftsprozesse und der IT-Prozesse miteinander zu verbinden. Dafür werden Diagrammtypen für die Daten- und Organisations*struktur* (bspw. Klassendiagramm) und für dessen dynamisches *Verhalten* bzw. Interaktion untereinander (bspw. Aktivitätsdiagramm, Kommunikationsdiagramm) bereitgestellt.

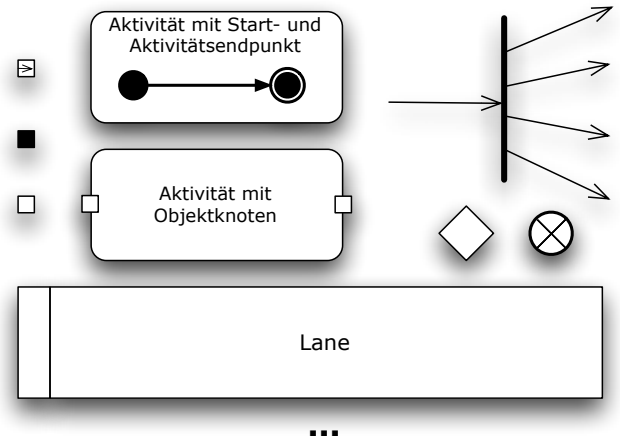

Abbildung 2.3: Elemente des UML Aktivitätsdiagramms - Auszug

UML ist im fachlichen Umfeld eher selten anzutreffen, da die Sprache zuviel „semantisches“ Beiwerk besitzt und es für die Darstellung vor Geschäftsprozessen passendere Sprachen (siehe

[1] IDS Scheer definiert hierfür verschiedene Modellierungssichten in dem sog. ARIS-Haus, welche aus Funktionssicht, Organisationssicht, Datensicht und Leistungssicht aufgeteilt sind.

[2] Object Management Group - Konsortium zur Entwicklung von Standards für die Objektorientierte Programmierung

EPK) existieren. Auf der technischen Ebene sind die jedoch häufiger vertreten, da sich alle Aspekte eines ausführbaren Prozesses mit den unterschiedlichen Diagrammtypen formalisieren lassen.

Abbildung 2.3 enthält einen Auszug aus den Elementen eines UML Aktivitätsdiagramms. Dieser Diagrammtyp stellt den Daten- und Kontrollfluss zwischen einzelnen Aktivitäten (vergleichbar zu Funktionen in EPK) dar, enthält außerdem Elemente zur Repräsentation von Daten und Exceptions sowie zur Gruppierung.

2.4.3 Business Process Modeling Notation - BPMN

Die Business Process Modeling Notation erlaubt es, vergleichbar zu UML, von der fachlichen bis zur technischen Ebene Geschäftsprozesse zu modellieren. BPMN wurde von der BPMI[3] entwickelt und wird momentan von der OMG gepflegt, da diese beiden Organisationen 2005 fusioniert sind. Im Gegensatz zu UML steht nicht ausschließlich die Möglichkeit die ebenenübergreifende Modellierung im Vordergrund, sondern vor allem die Verständlichkeit der Notation für den Fachanwender. Der Diagrammtyp von BPMN heißt „Business Process Diagram“ (BPD) und orientiert sich sehr stark an den Aktivitätsdiagrammen von UML. Die Notationselemente von BPD's lassen sich in vier Kategorien einteilen (siehe Abbildung 2.4).

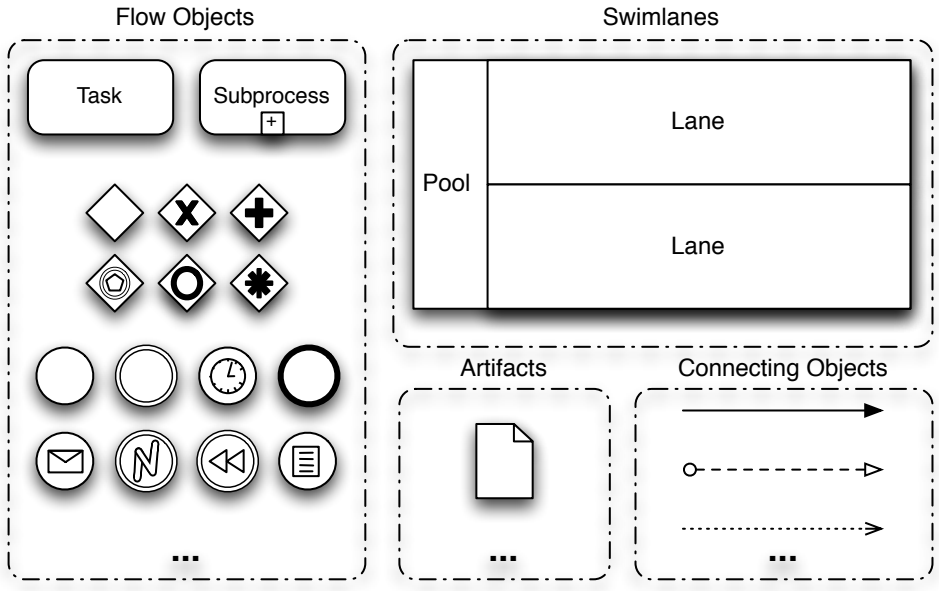

Abbildung 2.4: Business Process Modeling Notation - Auszug

Der Vorteil, für die Geschäftsprozessmodellierung mit BPMN, ist die Möglichkeit, einzelne Elemente und Teilaspekte weniger formal beschreiben zu müssen. Die Semantik von beispielsweise Gateways als Verzweigung ist zwar klar vorgegeben, jedoch bleibt es dem Prozessmodellierer überlassen auf welche Art die Bedingungen für die Verzweigungen ausgewählt werden. Dies lässt BPMN einen gewissen Spielraum bei der Modellierung, wodurch ein Modell Stück für Stück

[3]Business Process Management Initiative

stärker formalisiert werden kann, ohne zu einem Zeitpunkt aufgrund fehlender Bedingungen inkonsistent zu werden.

Mit der momentan in Beta v1 vorliegenden Version 2.0 wird BPMN um eine Ausführungssemantik erweitert [BPM09], die eine formalere Definition einzelner Bestandteile für die automatische Evaluierung voraussetzt.

2.5 SOA Modellierungsmethodik

Im Zusammenhang mit SOA wird für Geschäftsprozesse während der Ausführung eine gewisse Flexibilität gefordert. Das ergibt sich daraus, dass strategische oder taktische Änderungen in einem Unternehmen sich möglichst schnell in einem geänderten Geschäftsprozess auswirken sollen. Dies kann beispielsweise eine geänderte Produktlinie mit neuen Zulieferern sein, oder Organisationsänderungen die das ganze Unternehmen betreffen. Um diese Flexibilität zu erreichen müssen Fachbereich und IT-Bereich möglichst gut koordiniert werden, da sich Änderungen an Geschäftsprozessen über beide Bereiche erstrecken. Hierfür ist es hilfreich, wenn der IT-Bereich über fachliche Änderungen unterrichtet wird, und betroffene Aktivitäten im ausführbaren Prozess identifizieren kann kann. Dazu ist es notwendig, dass diese technischen Aktivitäten ihren fachlichen Aktivitäten (und vice versa) zugeordnet werden können [BBR10, BBR09].

Diese Problematik zu verstehen, wird exemplarisch anhand des in Kapitel 1 vorgestellten Anwendungsszenarios der Fachprozess und der ausführbare Prozess vorgestellt.

2.5.1 Fachprozess

Auf der fachlichen Ebene wird davon abstrahiert, wie der Geschäftsprozess auf der technischen Ebene durchlaufen wird. Es wird beschrieben, *welche Schritte* dafür notwendig sind, bzw. welche Personen oder Ressourcen daran beteiligt sind, um den Prozess zu abzuschließen.

Dementsprechend hoch ist idealer Weise auch der Abstraktionsgrad der Modellierungssprache von der technischen Implementierung. Fachliche Abläufe müssen vollständig, jedoch noch in einer durch den Fachbereich verständlichen Notation beschrieben werden können. Aus diesem Grund wird das Anwendungsszenario in den bereits vorgestellten EPK's modelliert.

Abbildung 2.5: Fachliches Prozessmodell des Anwendungsszenarios

Die einzelnen Funktionen beschreiben dabei analog zu der Wertschöpfungskette, die einzelnen Prozessschritte sehr textuell, wodurch die Übersichtlichkeit für den fachlichen Bereich erhalten bleibt.

2.5.2 Ausführbares Modell

Die Anforderung an die technische Ebene, also den eigentlichen ausführbaren Prozess, ist es nun zu beschreiben, *wie* die in dem fachlich modellierten Geschäftsprozess definierten einzelnen Ergebnisse erreicht werden können. Einerseits erfordern fachliche Aufgaben eine manuelle Bearbeitung (Human Tasks[4]), andererseits jedoch einen Aufruf eines Dienstes (Services) der diese Aufgabe durchführt.

Modellierungssprachen wie BPEL, können bzw. müssen sehr stark mit technischen Details angereichert sein. Diese sind für den Fachbereich eher unübersichtlich, da sie zwar den fachlich modellierten Geschäftsprozess ebenso beschreiben, jedoch für diesen sehr viele irrelevanten Informationen enthalten.

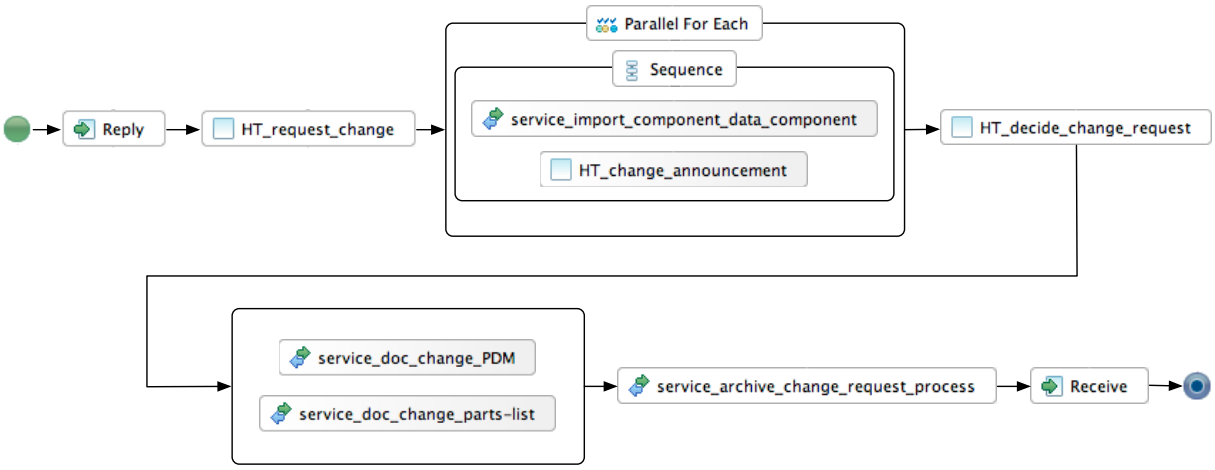

Abbildung 2.6: Ausführbarer Prozess des Anwendungsszenarios

Wie anhand des Fachprozesses und des ausführbaren Prozesses zu sehen ist, sind die Veränderungen, die durch den IT-Bereich an dem Fachprozess geschehen sind, schwer nachzuvollziehen. Die einzelnen Aktivitäten des Fachprozesses haben im ausführbaren Prozess unterschiedliche Bezeichnungen, oder wurden vollkommen durch technische Aktivitäten ersetzt. Ähnlich verhält es sich mit Verzweigungen des Kontrollflusses. Während in dem fachlichen Prozess keine Verzweigungen auftreten, sind diese im BPEL Prozess vorhanden (siehe Abbildung 2.6).

Dadurch entsteht eine *Verständnislücke* der jeweiligen Modelle zwischen Fach- und IT-Bereich.

Fachliche Anforderungen (bspw. die Festlegung von max. Durchlaufzeiten für einzelne Prozessschritte), die im Fachmodell meist textuell beschrieben werden, sind schlechte Vorgaben für die technische Implementierung. Die Implementierer des ausführbaren Prozesses besitzen nicht das nötige Domänenwissen um diese fachlichen Anforderungen zu verstehen.
Auf der anderen Seite kann der ausführbare Prozess für den Fachbereich vollkommen unverständlich sein. Die Veränderungen im Vergleich zum Fachmodell lassen ohne technisches Wissen

[4]Eine Human Task stellt eine Einbindung einer menschlichen Interaktion in einem Geschäftsprozess innerhalb von BPEL dar [RvdA08].

keine Rückschlüsse auf den ursprünglichen Prozess zu. Um diese Verständnislücke zu überwinden, müssen die Differenzen und die Zusammenhänge zwischen Fach- und Systemaktivitäten in Workshops mit Vertretern aus Fach- und IT-Bereich herausgefunden werden.
Dieses Vorgehen ist langwierig und unterstützt somit nicht die Anforderungen an Flexibilität innerhalb einer SOA.

2.5.3 Systemmodell und Systemprozess

Um diese Abstimmung in effizienter Weise zu unterstützen wird zwischen Fachbereich bzw. Fachmodell und IT-Abteilung bzw. ausführbarem Modell eine neue Modellebene, das *Systemmodell*, eingeführt [BBR09, BBR10]. Ziel des Systemmodells ist es, eine Zwischenschicht zu bilden, die der Fachbereich einerseits versteht und durch den IT-Bereich als „Vorlage“ für die Implementierung des späteren ausführbaren Prozesses verwendet werden kann. Diese „Vorlage“ wird vom Fachbereich als „lesbare“ Repräsentation des ausführbaren Prozesses verwendet, und wird im folgenden als *Systemprozess* bezeichnet.

Aktivitäten des Systemprozesses tragen bereits die Bezeichnungen des ausführbaren Prozesses. Auch der Kontrollfluss ist in beiden Modellen identisch. Der Unterschied ist jedoch, dass hier von der technischen Implementierung abstrahiert wird. Plattformspezifische Verfeinerungen finden nach wie vor bei der Erstellung des ausführbaren Prozesses statt. Die Abbildung der Systemprozessaktivitäten auf die Aktivitäten des ausführbaren Modells geschehen daher 1:1 über den Namen[5].

Die initiale Erstellung des Systemprozesses findet durch eine Kooperation von Fachbereich und IT-Bereich statt. Die anschließende Pflege soll von beiden Bereichen autark erfolgen können, da die zeitaufwändige Absprache mittels dieser Modellschicht umgangen werden soll.
Abbildung 2.7 zeigt den Systemprozess des Anwendungsszenarios in BPMN.

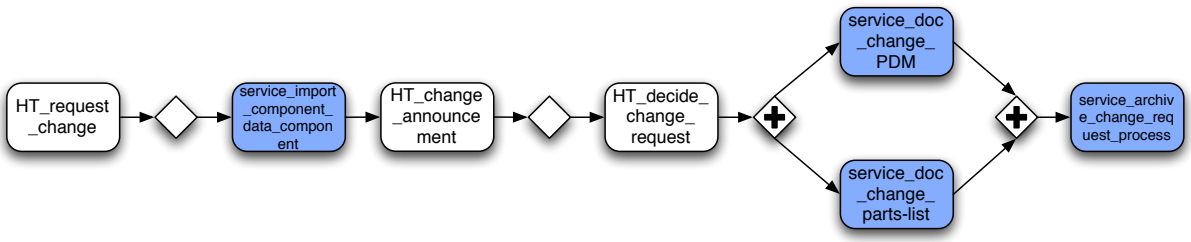

Abbildung 2.7: Systemprozess des Anwendungsszenarios

Im folgenden werden die einzelnen Umstrukturierungen beschrieben, um vom Fachmodell in Abbildung 2.5 zum Systemprozess in Abbildung 2.7 zu gelangen.

Das Stellen eines Änderungsantrags für ein oder mehrere Bauteile ① wird auf eine Human Task (HT_request_change) ⓐ abgebildet, da es sich hier um einen manuellen Schritt handelt und keine technische Repräsentation hat.

[5]Falls die Namenskonventionen im Systemmodell noch nicht entgültig sind, ist eine Abbildungstabelle denkbar.

Die Bewertung und Genehmigung einer Änderung ② besitzt auf der technischen Ebene zwei atomare Äktivitäten. Zum einen die Abfrage der technischen Details der betroffenen Bauteile (bspw. durch einen Service Aufruf - service_import_component_data_component) ⓑ, und zum anderen die wiederum manuelle Genehmigung der Änderung durch einen Techniker (HT_change_announcement) ⓒ. Das bedeutet für den Systemprozess, dass die entsprechende fachliche Aktivität auf der technischen Seite in zwei technische Aktivitäten aufgespalten wird. Diese beiden Schritte sollen für jedes betroffene Bauteil parallel ausgeführt werden, weshalb hier in BPMN ein Gateway eingesetzt wurde, das mit entsprechender Annotation für den ausführbaren Prozess auf eine Parallel-For-Each abgebildet wurde.

Die Entscheidung über die Art der Umsetzung ③ wird wiederum manuell getätigt, und wird auf eine Human Task (HT_decide_change_request) ⓓ abgebildet.
Für die Dokumentation der Entscheidung, welches Bauteile geändert wird, und welche Bauteile betroffen sind ④, werden zwei Services (service_doc_change_PDM, service_doc_change_partslist) ⓔ ⓕ aufgerufen, die die entsprechenden Veränderungen in den Datenbanken vornehmen. Diese beiden Serviceaufrufe sollen gleichzeitig geschehen, was durch ein AND-Gateway eingeleitet wird.
Das Umsetzen der Änderung ⑤ hat im Kontext dieses Prozesses keine direkte technische Entsprechung und taucht im Systemmodell nicht auf. Neu hinzu gekommen ist jedoch ein letzter Service Aufruf (service_archive_change_request_process) ⓖ, der die Ausführung des Prozesses im System dokumentiert.

2.5.4 Abbildungsmodell

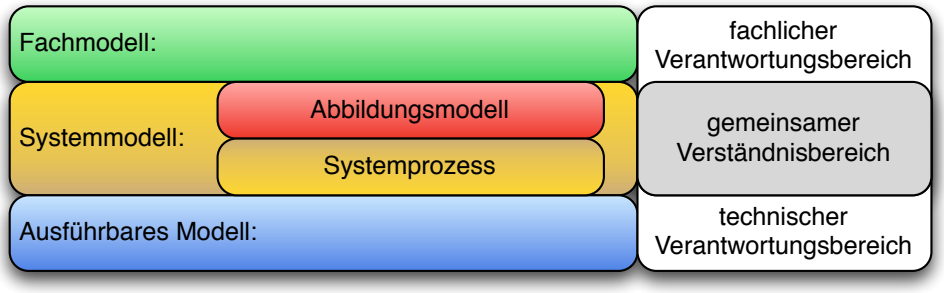

Abbildung 2.8: Abbildungsmodell und Systemprozess

Wie bereits beschrieben, werden die Systemprozessaktivitäten, nachfolgend Systemtasks genannt, 1:1 in den ausführbaren Prozess übernommen.
Da der Fachprozess sehr stark verändert wurde um zum Systemprozess zu gelangen, ist diese 1:1 Übernahme der Aktivitäten hierbei jedoch nicht möglich. Für die Dokumentation der Umstrukturierung des Fachmodells auf den Systemprozess, wird daher oberhalb des Systemprozesses eine neue Modellebene eingeführt, das *Abbildungsmodell* (siehe Abbildung 2.8). Damit besteht das Systemmodell einerseits aus dem Abbildungsmodell und andererseits aus dem Systemprozess. Diese Unterteilung ermöglicht die graphische Dokumentation der Überführungen der Fachaktivitäten auf die Systemtasks in der Systemmodellebene.

Im Zusammenhang mit dem Abbildungsmodell werden die Überführungen nachfolgend *Trans-*

formationen genannt, die im folgenden genauer beschreiben werden. Transformationen bestehen aus *Quellobjekten* (den Repräsentanten der Fachtasks), *Zielobjekten* (den Repräsentanten der Systemtasks) und den *Transformationsknoten*, die Operationen definieren, um diese Objekte untereinander zu überführen. Transformationsknoten werden im Abbildungsmodell durch Achtecke repräsentiert, welche eine textuelle Typbezeichnugn besitzen. Es werden 5 Transformationstypen eingeführt, die im folgenden formal beschrieben werden.

Umbenennen Durch die Umbenennen-Transformation wird eine Fachtask auf eine Systemtask abgebildet. Diese Operation fungiert auch im Fall der selben Bezeichnung von Fach- und Systemtask als 1:1 Abbildung von einer Task im Abbildungsmodell. Der Transformationsknoten besitzt daher eine eingehende und eine ausgehende Kante.

Regel 1
*Sei T_{Umb} ein Transformationsknoten mit dem Typ „Umbenennen“, #E die Anzahl der eingehenden Kanten und #A die Anzahl der ausgehenden Kanten. Dann gilt: #**E**=**1** und #**A**=**1***

Aufspalten Die Aufspalten-Transformation überführt eine Fachtasks in mehrere Systemtasks (beispielsweise wird eine fachliche Aktivität in einen Service Aufruf und einen manuellen Schritt zerlegt). Es handelt sich somit um eine 1:n Beziehung.

Regel 2
*Sei T_{Auf} ein Transformationsknoten mit dem Typ „Aufspalten“, #E die Anzahl der eingehenden Kanten und #A die Anzahl der ausgehenden Kanten. Dann gilt: #**E**=**1** und #**A**=**n**, $n>1$*

Zusammenfassen Bei der Zusammenfassen-Transformation werden mehrere Fachtasks zu einer Systemtask zusammengefasst. Dies hat den praktische Anwendung, dass beispielsweise zwei Aktivitäten im Fachprozess auf der technischen Seite durch einen zusammengehörigen Service realisiert werden. Hierbei handelt es sich um eine m:1 Beziehung zwischen Fach- und Systemtasks.

Regel 3
*Sei T_{Zus} ein Transformationsknoten mit dem Typ „Zusammenfassen“, #E die Anzahl der eingehenden Kanten und #A die Anzahl der ausgehenden Kanten. Dann gilt: #**E**=**m** und #**A**=**1**, $m>1$*

Einfügen Durch die Einfügen-Transformation wird dem Systemprozess eine Systemtask hinzugefügt, die keine direkte Entsprechung im Fachprozess besitzt.

Regel 4
*Sei T_{Einf} ein Transformationsknoten mit dem Typ „Einfügen“, #E die Anzahl der eingehenden Kanten und #A die Anzahl der ausgehenden Kanten. Dann gilt: #**E**=**0** und #**A**=**1***

Löschen Analog zur Einfügen-Transformation kann im Systemprozess eine Fachtask keine direkte Entsprechung besitzen. Falls die Zusammenfassen-Transformation in diesem Fall keinen Sinn macht, kann die entsprechende Task im Systemprozess gelöscht werden.

Regel 5
*Sei $T_{Lös}$ ein Transformationsknoten mit dem Typ „Löschen“, #E die Anzahl der eingehenden Kanten und #A die Anzahl der ausgehenden Kanten. Dann gilt: #**E=1** und #**A=0***

Aus dem Grund, dass das Systemmodell und somit das Abbildungsmodell die Komplexität für die Nachvollziehbarkeit der Änderungen eher verringern soll, existiert keine explizite n:m Beziehung. Diese kann bei Notwendigkeit mit den vorhandenen 5 Transformationen nachgebildet werden.

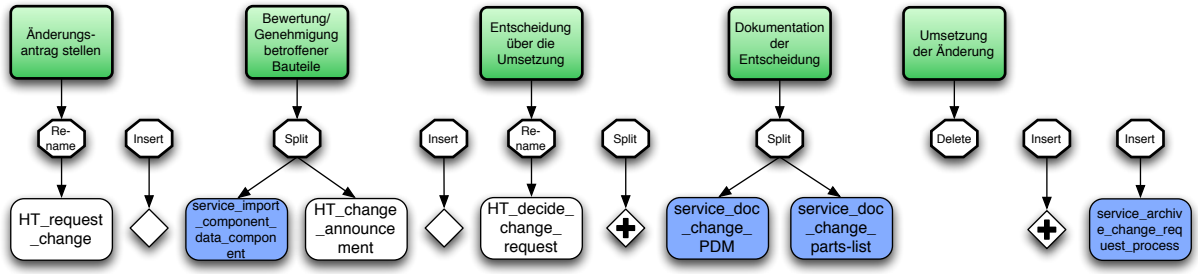

Abbildung 2.9: Abbildungsmodell des Anwendungsszenarios

Abbildung 2.9 zeigt die Abbildung des Fachprozesses auf den Systemprozess mit Hilfe der Transformationsknoten des Abbildungsmodells.

Das soeben beschriebene Anwendungsszenario besteht somit aus vier Ebenen, welche zusammengefasst in folgender Abbildung 2.10 einheitlich zu sehen sind. Im nachfolgenden Kapitel werden die oberen drei Ebenen genauer betrachtet.

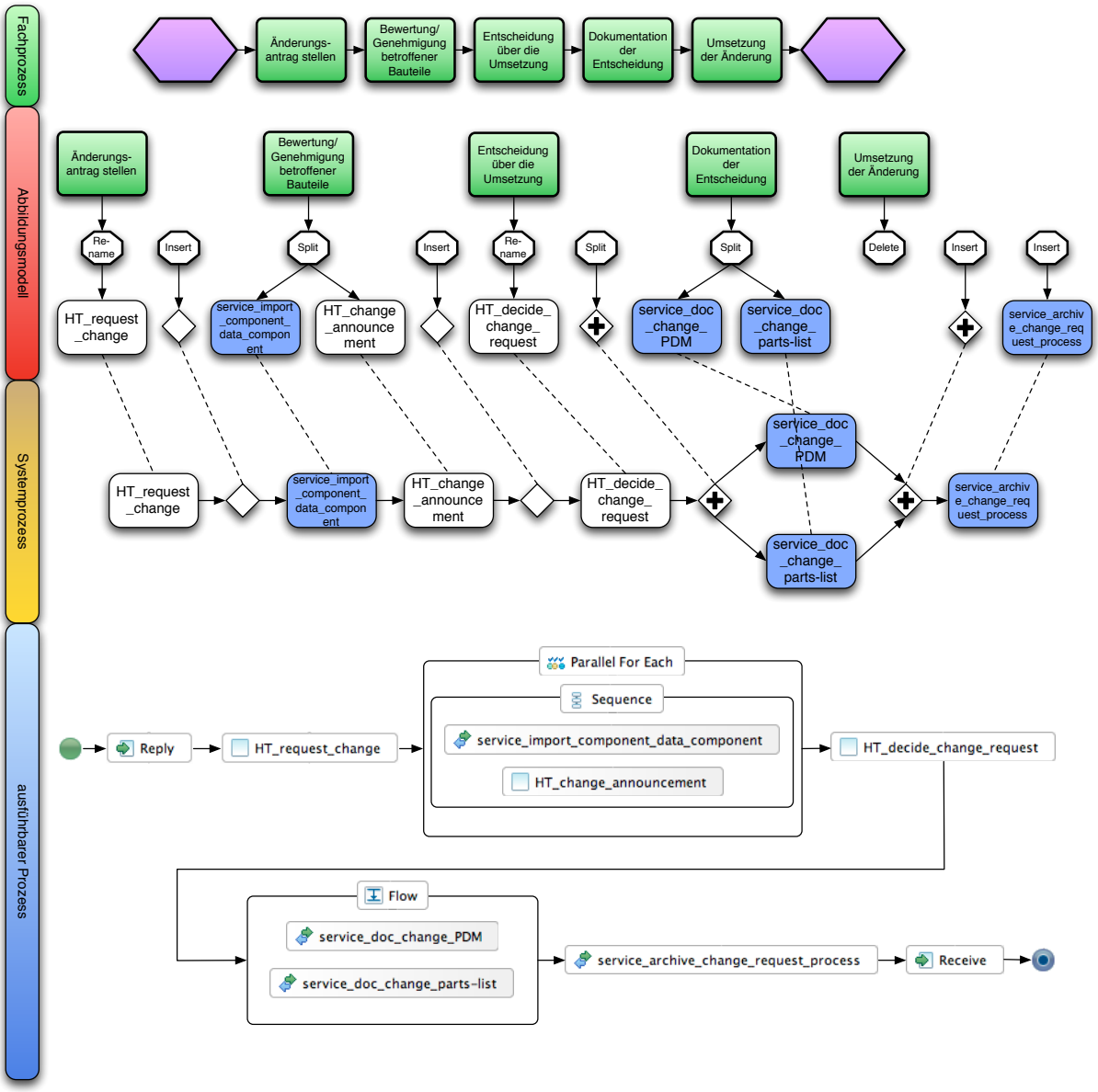

Abbildung 2.10: Gesamtes Anwendungsszenario

3

Vorgehensweisen zur Erstellung des Abbildungsmodells

Für die Überführung des fachlichen Prozesses in den Systemprozess werden in diesem Kapitel verschiedene Methoden betrachtet. Diese unterscheiden sich zum einen in der Reihenfolge der Erstellung der Modelle sowie in ihrem „Automatisierungsgrad“. Im Mittelpunkt steht das Abbildungsmodell, da es das Bindeglied zwischen Fach- und Systemprozess darstellt. Dieses Kapitel untersucht verschiedene Methoden zur Erstellung des Abbildungsmodells. Dabei wird insbesondere auf Vor- und Nachteile unterschiedlicher Vorgehensweisen sowie auf den Grad der (teil-) automatischen Generierung bestimmter Abbildungsmodell-Fragmente geachtet.
Methode 1 und 2 beschreiben allgemeine Lösungsansätze für die direkte Einbindung des Abbildungsmodells in das Systemmodell, wohingegen Methode 3 und 4 darauf aufbauend den Schwerpunkt auf eine möglichst gute Unterstützung des Systemmodellierers bei der Erstellung des Abbildungsmodells legen. Dazu werden die (teil-)automatisierbaren Bestandteile beschrieben und eingehend auf Umsetzbarkeit und Aufwand für den Systemmodellierer sowie Darstellungsart von Informationen untersucht.

3.1 Methode 1 - Beginnen mit Systemmodell

Modellierung des Fachprozesses ⓪: Voraussetzung für alle Methoden ist ein initial modellierter Fachprozess durch den Fachbereich. Es wird vorausgesetzt, dass siech dieser während der Erstellungsphase nicht verändert.
Das Ziel ist es, das Systemmodell, also Systemprozess mit Abbildungsmodell zu erstellen. Die Reihenfolge in der dies geschieht ist das hauptsächliche Unterscheidungsmerkmal der verschiedenen Methoden. Der Systemprozess beschreibt in einer für den Fachbereich verständlichen Notation vollständig den ausführbaren Prozess. Der Inhalt sowie die Struktur orientiert sich an der des Fachprozesses, muss jedoch weiter detailliert und ausreichend formal beschreiben werden um gleichzeitig eine Basis für die technische Umsetzung zu bilden.
Das Abbildungsmodell beschreibt hingegen graphisch die Beziehungen zwischen den Tasks des Fach- und Systemprozesses.

Bei der Erstellung dieser beiden Modelle soll nun untersucht werden, in welche Reihenfolge an die Modellierung herangegangen werden kann und welche Vor- und Nachteile sich daraus ergeben. Es kann des weiteren unterschieden werden, welche Schritte explizit von einem Benutzer durchgeführt werden müssen, und welche Teile generiert werden können. Das Resultat wird in jeder der einzelnen Methoden ein vollständiges und syntaktisch sowie semantisch korrektes Systemmodell sein.

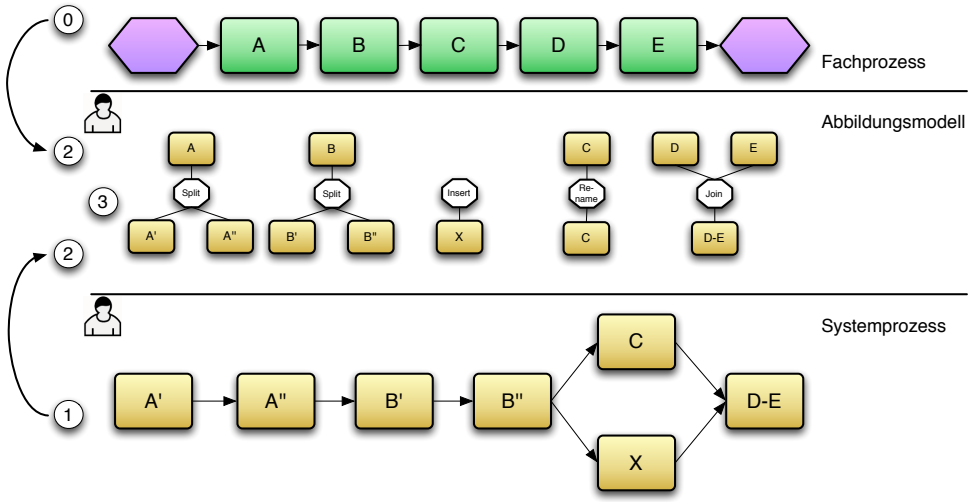

Abbildung 3.1: Übersicht Methode 1

Für dieses Kapitel werden zur Übersichtlichkeit die Beschreibungen der Tasks des Anwendungsszenarios durch Buchstaben ersetzt. Die Transformationen um die Änderungen des Fachprozesses im Abbildungsmodell zu dokumentieren wurden ebenfalls verändert, um alle möglichen Transformationen zu beinhalten.

In dieser ersten Methode wird ausgehend vom Fachprozess der Systemprozess frei modelliert. Dafür kann mit Hilfe des Fachprozesses der Systemprozess mit seinen Tasks, dem Prozessablauf und allen weiteren ausführungsrelevanten Aspekten modelliert werden. Um die bidirektionale Nachvollziehbarkeit zwischen Fach- und Systemtasks sicher zu stellen, werden anschließend im

Abbildungsmodell die Beziehungen zwischen diesen Tasks explizit gespeichert. Dafür müssen alle Tasks aus den jeweiligen Modellebenen importiert und durch Transformationsknoten aufeinander abgebildet werden.

Abbildung 3.1 zeigt das Vorgehen zur Erstellung eines entsprechenden Abbildungsmodells, welches im folgenden beschreiben wird.

3.1.1 Erstellung des Systemprozesses und des Abbildungsmodells

Erstellung des Systemprozesses ①: Der Systemprozessmodellierer erstellt ausgehend von technischen Anforderungen den Systemprozess. Diese Anforderungen sind im einfachsten Fall Namenskonventionen, welche für die spätere Implementierung relevant sind, und reichen bis zur Detaillierung einzelner Attribute technischer Tasks. Diese Informationen ist auf Ebene der Fachmodellierung nicht nötig oder nicht bekannt, teilweise sogar absichtlich vage und unformal gehalten.
Nach der Erstellung der Systemprozesstasks wird der Kontrollfluss zwischen den Tasks inklusive evtl. vorkommenden Verzweigungen festgelegt. Der Systemprozess ist nun in einer plattformunabhängigen Darstellung, welche als Spezifikation an den IT-Bereich zur Implementierung gegeben werden kann. Letztere bildet zusammen mit dem Fachprozess die Grundlage für die Modellierung des Abbildungsmodells.

Erstellung des Abbildungsmodells ②: Die Tasks aus Fach- und Systemprozess können nun in das Abbildungsmodell kopiert werden. Dafür werden sie manuell aus dem jeweiligen Modell exportiert und im Abbildungsmodell als Kopie des eigentlichen Objekts importiert. Dieser Schritt wird vom Systemmodellierer manuell, einzeln für den Fachprozess und für den Systemprozess durchgeführt. Anschließend befinden sich die Tasks aus dem Fachprozess und dem Systemprozess im Abbildungsmodell.

Bearbeitung des Abbildungsmodells ③: Nachdem nun alle Tasks im Abbildungsmodell dokumentiert sind, müssen sie nun in Beziehung gesetzt werden, d.h. die Fachtasks werden über Transformationsknoten mit den Systemtasks verbunden. Dies wird manuell vom Systemprozessmodellierer übernommen. Mit Hilfe der Transformationsknoten können durch die explizite graphische Modellierung die Zielobjekte aus den Quellobjekten abgeleitet werden. Die Beschriftung des jeweiligen Transformationsknotens entspricht dabei der Operation, durch welche die Systemtask aus der Fachtask hervor geht.

3.1.2 Fazit:

In Methode 1 wird der Systemprozess ohne Einschränkungen frei modelliert. Die Modellierung des Systemprozesses ist in dieser Methode der erste Schritt für die Erstellung des Systemmodells und die Erstellung des Abbildungsmodells ist deshalb mit vergleichsweise geringem Aufwand möglich, da die Fach- und Systemtasks in den jeweiligen Modellen bereits vorliegen und nur noch werden müssen.

Für die Integration dieser Methode in ein bestehendes Werkzeug wird sehr wenig Zusatzfunktionalität benötigt. Die Beziehungen der Objekte zwischen Fachprozess und Abbildungsmodell sowie zwischen Abbildungsmodell und Systemprozess sind implizit über den Namen gegeben, können alternativ aber auch anders referenziert werden. Es wird kein eigener Modelltyp für das Abbildungsmodell benötigt, da es entweder den Modelltyp des Fachprozesses oder den des Systemprozesses verwenden kann. Auch die Transformationsknoten sind nicht zwangsläufig an die in dieser Arbeit definierten Symbole gebunden.

Die Freiheit in der Modellierung kann unter gewissen Bedingungen jedoch auch von Nachteil sein. Zum einen wird ein Großteil der Verantwortung für die korrekte Modellierung der einzelnen Teilaspekte dem Systemmodellierer übergeben. Es muss während der Erstellung des Abbildungsmodells darauf geachtet werden, dass das Systemmodell alle Tasks des Fachprozesses durch das Abbildungsmodell abbildet.
Da der Systemprozess als erstes erstellt wird, wirken sich Fehler in der Modellierung direkt auf das Abbildungsmodell aus und fallen oft erst hier auf. Wenn beispielsweise im Systemprozess eine Task nicht modelliert wurde, erkennt der Systemmodellierer erst bei der Definition der Beziehungen im Abbildungsmodell diesen Fehler. Dies ist nicht nur eine mögliche Fehlerquelle, sondern kann auch einen erheblichen Mehraufwand dies zu korrigieren bedeuten.

Im folgenden werden die Vor- und Nachteile der Methode 1 zusammenfassend aufgelistet:

+ Freie Modellierung des Systemprozesses.
+ Bei Erstellung des Abbildungsmodells sind alle Tasks bereits definiert.
- Manuelles Kopieren aller Tasks zwischen den Modellen.
- Die Verantwortung für Vollständigkeit der Tasks auf den 3 Ebenen liegt beim Systemmodellierer.
- Manuelle Modellierung der Transformationsknoten zwischen Quell- und Zielobjekten des Abbildungsmodells.

3.2 Methode 2 - Beginnen mit Abbildungsmodell

Methode 2 beschreibt die Erstellung des Systemprozesses ausgehend vom Abbildungsmodell. Das Abbildungsmodell wird als erstes erstellt und anschließend aus den dort definierten Beziehungen der Systemprozess modelliert. Der Systemprozess wird somit nicht frei modelliert, sondern startet auf Basis der durch das Abbildungsmodell spezifizierten Taskmenge. D.h. Systemmodellierer ergänzen den Systemprozess ausschließlich um den Kontrollfluss.

Abbildung 3.2 zeigt die Vorgehensweise in Methode 2.

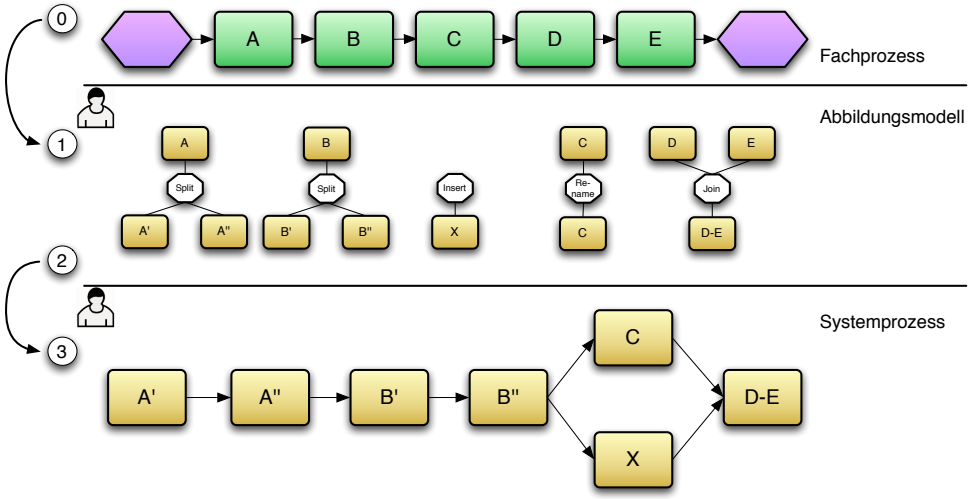

Abbildung 3.2: Übersicht Methode 2

3.2.1 Erstellung des Systemprozesses und des Abbildungsmodells

Modellierung des Fachprozesses ⓪: Analog zu Methode 1 wird hier ein modellierter Fachprozess vorausgesetzt.

Erstellung des Abbildungsmodells ①: Im ersten Schritt müssen alle Tasks aus dem Fachprozess in das Abbildungsmodell übertragen werden. Dies wird realisiert, indem der Systemmodellierer alle Fachtasks manuell importiert und diese ggf. in die Metamodellsprache des Abbildungsmodells konvertiert. Nun muss für die jeweilige Fachtask festgelegt werden, auf welche Systemtask(s) diese abgebildet werden sollen. Die Transformationsknoten werden im nächsten Schritt analog zu Methode 1 mit den Quell- und Zielobjekten verbunden. Die Menge an Systemprozesstasks muss im Abbildungsmodell vom Systemmodellierer vollständig spezifiziert werden, da auf dessen Basis im nächsten Schritt der Systemprozess erstellt wird.

Erstellung des Systemprozesses ②: Die Systemtasks des Abbildungsmodells werden vom Implementierer manuell in den Systemprozess übertragen. Diese kann bei einem möglichen Metamodellwechsel über einen manuellen Export bzw. Import geschehen, oder bei gleicher Metamo-

dellsprache zwischen Abbildungsmodell und Systemprozess durch kopieren gelöst werden.

Bearbeitung des Systemprozesses ③: Der Systemprozess wird nun vervollständigt. Dazu werden die einzelnen Tasks des Systemprozesses mit ausfühungsrelevanten Informationen, wie etwa Attributen oder Datenobjekten, verfeinert. Außerdem wird in dieser Phase der Kontrollfluss zwischen Tasks und Strukturknoten modelliert.

3.2.2 Fazit:

Ein Vorteil dieser Methode ist der direkte Bezug zu den Fachtasks während der Modellierung des Abbildungsmodells. Diese sind daher bereits bei der Definition der Systemtasks vollständig vorhanden. Dadurch erhält der Systemmodellierer einen Überblick über die noch zu erstellenden Beziehungen, da Fachtasks ohne einen zugehörigen Transformationsknoten zu diesem Zeitpunkt noch nicht bearbeitet wurden. Die Integrationsfähigkeit dieser Methode in bestehende Werkzeuge ist vergleichbar mit der von Methode 1. Es wird keinerlei Zusatzfunktionalität benötigt, da mit bestehenden Mitteln alle Aspekte des Systemmodells abgebildet werden können. Das liegt auch daran, dass in dieser Methode analog zu Methode 1 jeder Arbeitsschritt manuell vom Systemmodellierer selbst durchgeführt werden muss. Die Nachteile dieser Methode orientieren sich ebenfalls an denen von Methode 1. Der Systemmodellierer hat hier auch die Verantwortung alle Tasks des Fachprozesses vollständig auf die des Systemprozesses abzubilden. Fehlerhaft modellierte Transformationen werden oft erst im Systemprozess erkannt.
Ein weiterer Nachteil ist der Entwurf des Systemprozesses über das Abbildungsmodell. Der Vorteil, dass man permanent die Fachtaskmenge bei der Modellierung der Systemtasks im Abbildungsmodell zur Verfügung hat, wird dadurch aufgehoben, dass dort kein Bezug zum Kontrollfluss vorhanden existiert. Dieser wird Abbildungsmodell nicht modelliert. Die Abbildung der Tasks aufeinander wird dadurch zwar nicht erschwert, jedoch benötigt der Systemmodellierer bei einigen Transformationen Wissen über die " ´Umgebung“ in der sich eine Fachtask befindet. Für die Abbildung auf eine Systemtask kann es von Bedeutung sein, welche eingehenden Kanten (inkl. Bedingungen) diese hat, bzw. in welchem Zweig einer Verzweigung sich diese befindet.

Zusammenfassend lässt sich diese Methode folgendermaßen bewerten:

+ Nur der Kontrollfluss muss im Systemprozess eingezeichnet werden.
- Keine freie Modellierung des Systempozesses
- Definition der Systemtasks über das Abbildungsmodell.
- Manuelles Kopieren aller Tasks zwischen den Modellen.
- Manuelle Modellierung der Transformationsknoten zwischen Quell- und Zielobjekten des Abbildungsmodells.
- Die Verantwortung für Vollständigkeit der Tasks auf den 3 Ebenen liegt beim Systemmodellierer.

In den folgenden Methoden werden die Vorteile von Methode 1 und 2 kombiniert, während die Nachteile dieser minimiert werden.

3.3 Methode 3 - Start mit Kopie des Fachprozess

In dieser Methode wird analog zu Methode 1 zuerst der Systemprozess erstellt. Dafür wird der Fachprozess in den Systemprozess kopiert und im Anschluss daran das Abbildungsmodell generiert. Ziel dieser Methode ist es einerseits den Ersteller des Systemprozesses bei dessen Modellierung zu unterstützen, und andererseits das Abbildungsmodell weitgehend automatisch zu generieren. Hierzu werden die Bestandteile des Fach- und Systemprozesses verglichen, um aus deren Differenz die Informationen für die Erstellung des Abbildungsmodells zu gewinnen.

Abbildung 3.3 zeigt die Vorgehensweise in dieser Methode.

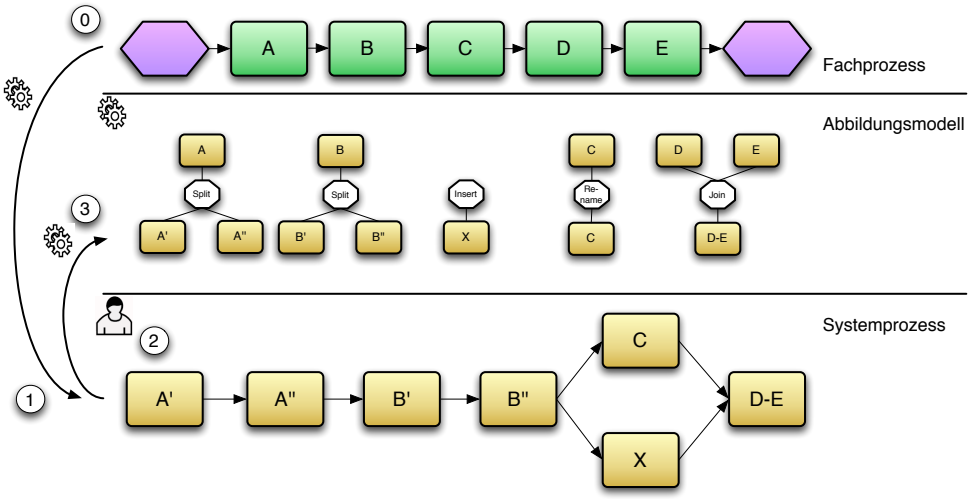

Abbildung 3.3: Übersicht Methode 3

3.3.1 Erstellung des Systemprozesses und des Abbildungsmodells

Modellierung des Fachprozesses ⓪: Analog zu Methode 1 wird auch hier ein modellierter Fachprozess vorausgesetzt.

Kopie des Fachprozess in das Systemmodell ①: Im ersten Schritt wird der Fachprozess in das Systemmodell überführt. Dazu werden Kopien aller Objekte inklusive deren Attribute im Systemprozess erstellt. Automatisierbare Schritte werden in Abbildung 3.3 mit zwei Zahnrädern (⚙) visualisiert, manuelle Schritte hingegen durch (). Bei einer automatisierten Kopie muss sichergestellt werden, dass beispielsweise Funktionen eines in EPK modellierten Fachprozesses in Tasks des BPMN-Systemmodells konvertiert und die entsprechenden Attribute übernommen werden. An dieser Stelle wird vorausgesetzt, dass die Konvertierung von dem jeweiligen Modellierungswerkzeug angeboten bzw. unterstützt wird.

Wenn im Fachprozess Verzweigungen auftreten, bei denen die semantische Bedeutung nicht ersichtlich ist, muss im Systemprozess entschieden werden, welche Art von Strukturknoten an dieser Stelle eingefügt werden soll. Diese Entscheidung muss vom Systemprozessmodellierer getroffen

werden. Beispielsweise werden zwei ausgehende Pfeile in BPMN, als auch in UML (Aktivitätsdiagramme) semantisch als UND-Verknüpfung interpretiert, wo im Fachprozess jedoch eine ODER-Verknüpfung gemeint ist. Solche Anpassungen muss der Systemmodellierer durchführen, wobei er dabei durch Konsistenzanalysen (vgl. Kapitel 6) unterstützt werden kann.

Bearbeiten des Systemprozesses ②: Der Systemprozess besteht nun aus einer Menge kopierter Tasks aus dem Fachprozess. Der Systemprozessmodellierer spezifiziert nun den Systemprozess. Hierzu wird der Prozess stark umstrukturiert. Einerseits werden Tasks zusammengefasst, falls sie im ausführbaren Prozess einen zusammengehörigen Service darstellen. Andererseits können sie auch aufgespalten werden, um einen für die technische Umsetzung zu allgemein beschriebenen Task in mehrere Tasks aufzuteilen. Da aus dem Systemprozess und dem Fachprozess das Abbildungsmodell abgeleitet werden soll, ist es nötig, dass alle Informationen für die Abbildung der Funktionen auf die Aktivitäten eindeutig zugeordnet werden können.
Beispielsweise kann eine fehlende Aktivität im Systemprozess entweder gelöscht oder zusammengefasst worden sein. Das ist durch den reinen Vergleich der Modelle auf Differenzen nicht festzustellen. Um diesen Konflikt aufzulösen, müssen die Änderungen, welche beim Bearbeiten des Systemprozesses angewandt wurden, protokolliert werden.
Dieses Problem macht es nötig, dass der Bearbeiter des Systemprozesses in seinem Freiheitsgrad für die Modellierung eingeschränkt wird. Änderungen werden nur über sog. Änderungsoperationen erlaubt, welche während der Änderung des Systemprozesses entscheiden, welche semantische Bedeutung diese Operation hat. Nachdem eindeutig ist, welche Änderung gemeint ist, muss für die spätere Generierung des Abbildungsmodells jede Änderung zusätzlich protokolliert werden.

Es existieren 5 mögliche Änderungsoperationen, mit denen sich alle Änderungen im Systemprozess durchführen lassen. Im folgenden wird zur Unterscheidung mit den Bezeichnungen der Transformationsknoten des Abbildungsmodells die Abkürzung „CO“ für „Change Operation“ verwendet

- CO_Einfügen: Im Systemprozess wird eine Task eingefügt
  (vgl. Abb. 3.4a)
- CO_Löschen: Im Systemprozess wird eine Task gelöscht
  (vgl. Abb. 3.4b)
- CO_Umbenennen: Im Systemprozess wird eine Task Umbenannt
  (vgl. Abb. 3.4c)
- CO_Zusammenfassen: Im Systemprozess werden mehrere Tasks zu einer zusammengefasst
  (vgl. Abb. 3.4d)
- CO_Aufspalten: Im Systemprozess wird eine Task in mehrere Tasks aufgespalten
  (vgl. Abb. 3.4e)

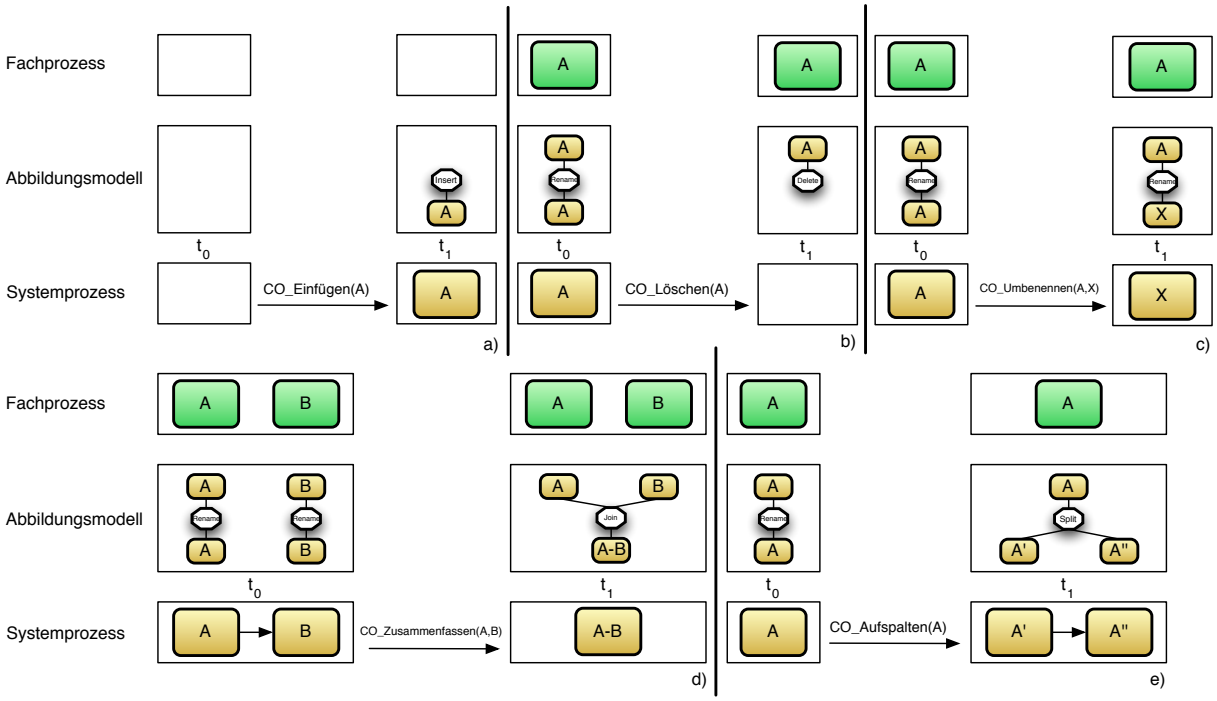

Abbildung 3.4: Änderungsoperationen auf dem Systemprozess

Generieren des initialen Abbildungsmodells ③: Bevor das Abbildungsmodell generiert werden kann, muss untersucht werden, wann die Änderungsoperationen angewendet werden sollen. Es existieren dafür zwei mögliche Zeitpunkte. Zum einen nachdem der Systemmodellierer den kompletten Systemprozess modelliert hat (alle Änderungsoperationen zusammen[als Liste]), oder zum anderen nach jeder einzelnen ausgeführten Änderungsoperation.

Variante 1 - Änderungsliste anwenden: Die ausgeführten Änderungsoperationen müssen dafür protokolliert werden. Die genaue Speicherung kann von Werkzeug zu Werkzeug variieren und ist für die Generierung nicht von Bedeutung. Wichtig ist nur, dass die Menge der Änderungen vollständig ist und in der richtigen Reihenfolge vorliegt, da nicht alle Änderungsoperationen kommutativ anwendbar sind.
Wie in Abbildung 3.4 zu sehen, wird das Abbildungsmodell erst nach dem Systemprozessbearbeitung erstellt. In dem anfangs leeren Abbildungsmodell werden zunächst mittels Umbenennen-Transformationen 1-zu-1 Beziehungen zwischen den Fachtasks und den Systemtasks generiert. Diese Transformationen dienen nun als Basis für die Folge von Änderungen aus dem Systemprozess, welche nach festgelegten Regeln nacheinander auf das Abbildungsmodell angewendet werden. Der Nachteil der sich daraus ergibt, ist, dass Fehler bei der Anwendung dieser „Änderungsliste“ (vgl. Abbildung 3.5) auftreten können, erst am Ende der eigentlichen Generierung aufgelöst werden können. Eine mögliche Situation, in welcher der Systemmodellierer manuell eingreifen muss, wäre beispielsweise das Zusammenfassen zweier Tasks, und das nachträgliche

Aufspalten. Da hierfür die Generierung unterbrochen werden muss führt das zu einem (evtl. erheblichen) Zeitverlust.

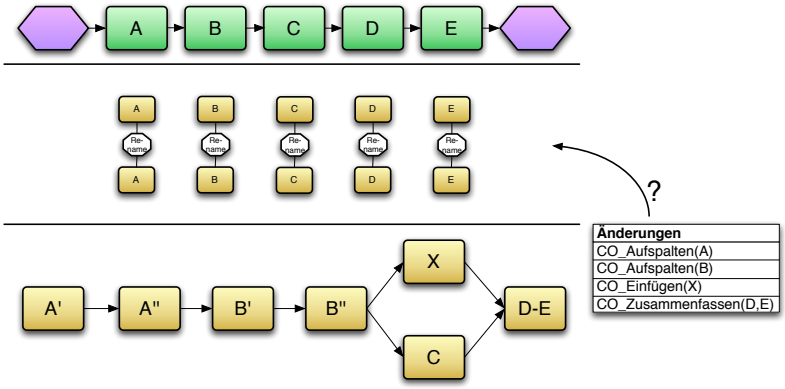

Abbildung 3.5: Initiales Abbildungsmodell, bestehend aus Umbenennen-Transformationen

Variante 2 - Direktes Anwenden von Änderungsoperationen: Als Ausgangspunkt dient wieder ein Abbildungsmodel, welches mit 1-zu-1 Beziehungen zwischen Fach- und Systemtasks generiert wird. Dies geschieht hier jedoch abweichend von Abbildung 3.3 parallel zu Erstellungsschritt ①, da es bereits in der Änderungsphase des Systemprozesses vorliegen muss. Die ausgeführten Änderungsoperationen wirken sich zeitgleich im Systemprozess und im Abbildungsmodell aus. Der Vorteil dieser Methode ist es, dass im Systemprozess, und somit im Werkzeug keine Änderungstabelle gespeichert werden muss. Jede Änderung spiegelt sich sofort im Abbildungsmodell wieder, wodurch Fach- und Systemprozess sowie das Abbildungsmodell während der Erstellungsphase permanent konsistent bleiben. Falls bei einer Umstrukturierung Konflikte auftreten, müssen diese vom Bearbeiter sofort aufgelöst werden, um mit der Anpassung des Systemprozesses fortzufahren.

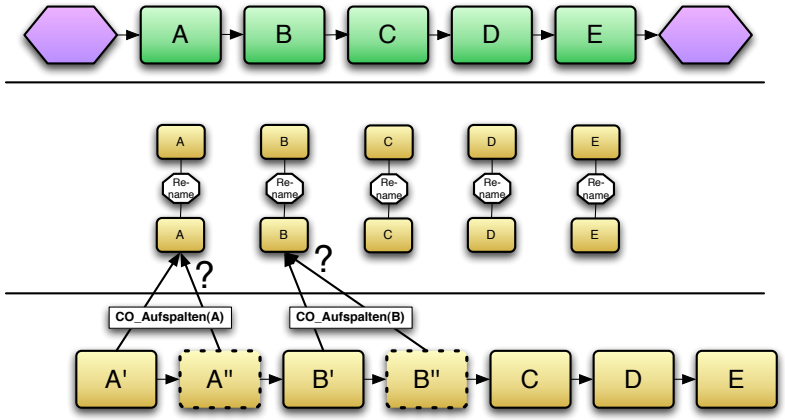

Abbildung 3.6: Aufspalten eines Systemtasks im Systemprozess

In den beiden Varianten, welche den Zeitpunkt festlegen, wann Änderungen des Systemprozesses im Abbildungsmodell angewendet werden können, dienen Umbenennen-Transformationen als

Grundlage für alle folgenden Änderungen. Wie Abbildung 3.4 zu entnehmen ist, ist jede Änderungoperation, welche auf einer Umbenennen-Transformation angewendet wird, klar definiert und ohne zusätzliche Informationen vom Bearbeiter anwendbar. Ausgehend davon wird untersucht, welche Änderungen im Systemprozess im Abbildungsmodell potentielle Problemfaktoren sein können.

3.3.2 Änderungsoperationen und deren Auswirkung auf das Abbildungsmodell

Im Folgenden werden die Änderungen des Abbildungsmodells nach den unterschiedlichen Änderungsoperationen im Systemprozess beschreiben. Zeitpunkt t0 legt den Zustand des Systemprozesses und des Abbildungsmodells fest, bevor die spezifische Änderungsoperation angewendet wurde. In der bereits vorgestellten Variante 1 bezieht sich t1 auf den Zeitpunkt, an dem alle Veränderungen im Systemprozess abgeschlossen sind (vollständige Änderungsliste vorhanden), während bei Variante 2 t1 die gerade abgeschlossene Änderungsoperation im Systemprozess bezeichnet.

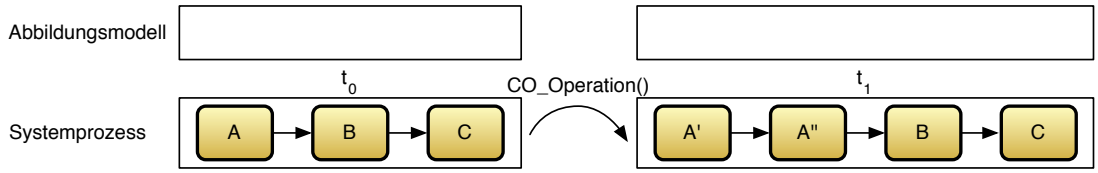

Abbildung 3.7: Übersicht über die Transformationsregeln

Wie bereits erwähnt ist eine Änderungsoperation auf einer Umbenennen-Transformation eindeutig zuzuordnen. Dies verhält sich gleichfalls mit der Einfügen- sowie der Löschen-Transformation. Diese drei Transformationen haben gemeinsam, dass sie jeweils nur eine Task beeinflussen, und dadurch leicht ihr Verhalten bestimmt werden kann. Komplizierter verhält es sich jedoch mit der Aufspalten- und Zusammenfassen Transformation.

Änderungsoperation(CO_Aufspalten): Beim Aufspalten werden aus einer Task im Systemprozesse, und somit auch im Abbildungsmodell, mehrere Tasks erstellt. Das besondere daran ist, dass für das Abbildungsmodell keine n:m Beziehungen definiert sind (vgl. Kapitel 2.5.4). Im Fall einer aus n Fachtasks zu zusammengefassten Systemtask (siehe Regel 3.11), welche während der Bearbeitung des Systemprozesses nun aufgespalten werden soll, muss diese n:m Beziehung aufgelöst werden. Das muss zu einem geeigneten Zeitpunkt durch den Systemmodellbearbeiter geschehen. Der Einfachheit halber soll in diesem Beipiel angenommen werden, dass nur 2 Fachtasks zusammengefasst wurden. Man kann dies theoretisch verallgemeinern, wobei für n Fachtasks, n+1 (sinnvolle) Kombinationen entstehen. Wie in Abbildung 3.11 zu sehen ist, stehen für die Auflösung dieses Spezialfalls mit 2 Fachtasks 3 Möglichkeiten zur Auswahl.
Alternative 1 und 2 ordnen die aufgespaltenen Systemtasks jeweils einer Fachtask zu, wohingegen in Alternative 3 die Fachtasks durch das Abbildungsmodell formal gelöscht werden und die aufgespaltenen Systemtasks neu eingefügt werden. Der zuständige Bearbeiter muss nun entscheiden, welche Alternative in dem jeweiligen Zusammenhang semantisch Sinn ergibt.

Task A wird in allen 4 Regeln im Systemprozess durch die „CO_Aufspalten“ aufgespalten, wodurch im Abbildungsmodell neue Systemtasks an den betroffenen Transformationen eingefügt werden müssen.

Regel A1: Die Umbenennen-Transformation wird zu einer Aufspalten-Transformation. Dies stellt den Basisfall auf einem neu generierten Abbildungsmodell dar.

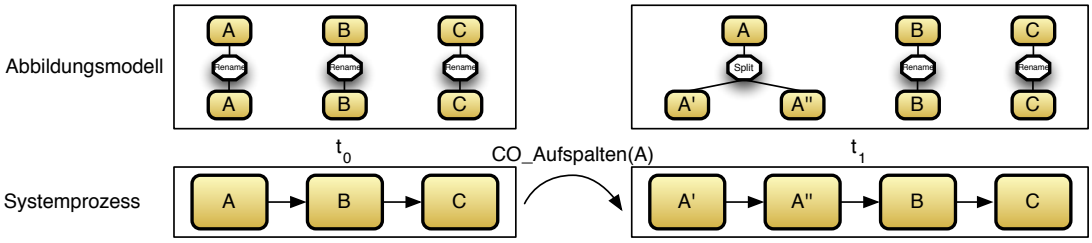

Abbildung 3.8: A1: Aufspalten Regel 1

Regel A2: Eine Einfügen-Transformation ist vom Aufspalten betroffen. Nach der Definition der Einfügen-Transformation in Kapitel 2.5.4 kann diese nur eine Systemtask besitzen. Daher muss eine neue Transformation erzeugt werden.

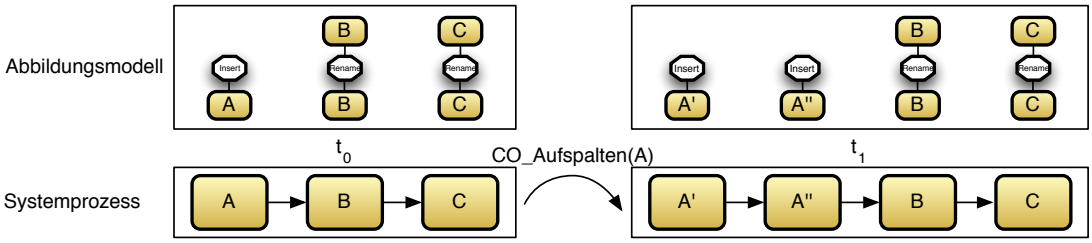

Abbildung 3.9: A2: Aufspalten Regel 2

Regel A3: Eine bereits aufgespaltene Task wird weiter aufgespalten. Die Aufspalten-Transformation bekommt eine neue Systemtask „angehengt“.

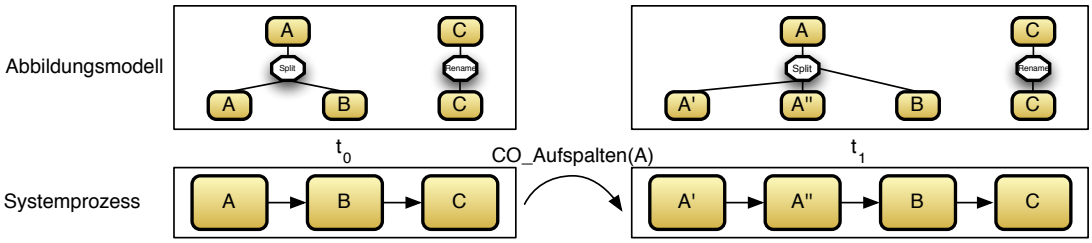

Abbildung 3.10: A3: Aufspalten Regel 3

Regel A4: Eine bereits zusammengefasste Task wird in der Bearbeitungsphase des Systemprozesses wieder aufgespalten. Der Modellierer muss nun entscheiden, wie dies im Abbildungsmodell aufgelöst werden soll.

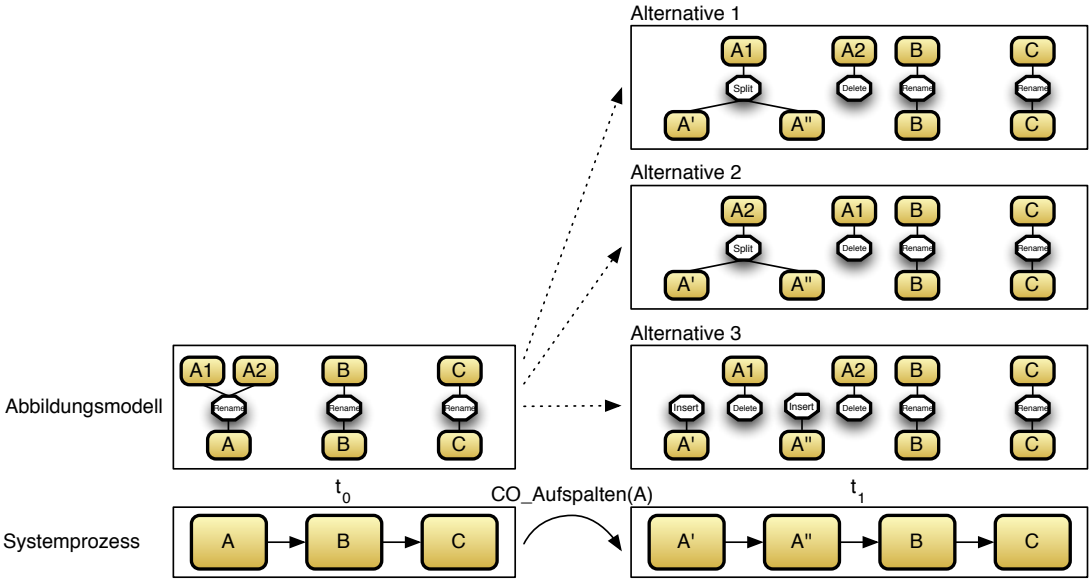

Abbildung 3.11: A4: Aufspalten Regel 4

Änderungsoperation(CO_Einfügen): Eine Task X wird im Systemprozess neu eingefügt.

*Regel E1:*Eine neue Task X wird im Abbildungsmodell mittels einer Einfüge-Transformation eingefügt. Hierbei sind keine weiteren Tasks, oder Transformationen betroffen.

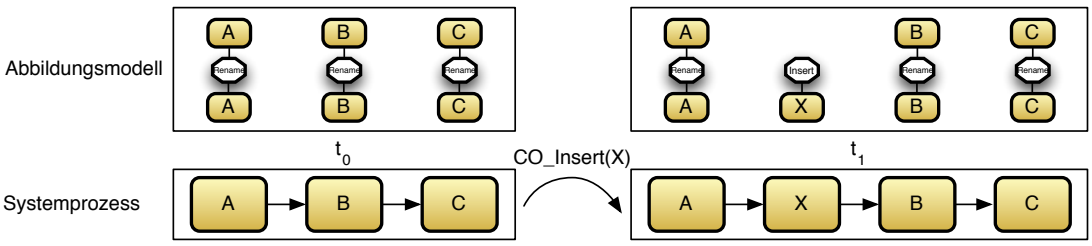

Abbildung 3.12: E1: Einfügen Regel 1

Änderungsoperation(CO_Löschen): Eine Task B wird aus dem Systemprozess gelöscht.

*Regel L1:*Die Umbenennen Transformation, in welcher die Task gelöscht wurde, wird zu einer Delete-Transformation.

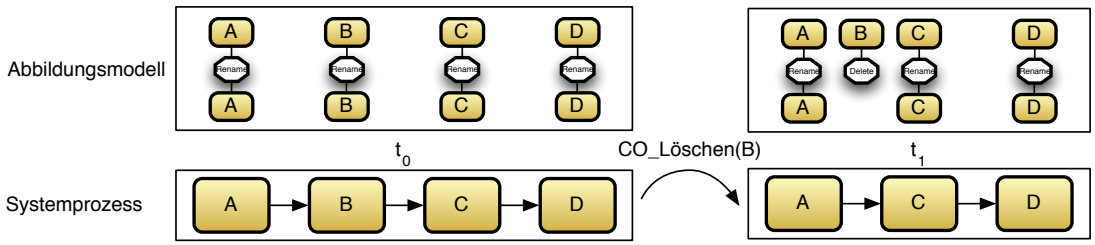

Abbildung 3.13: L1: Löschen Regel 1

*Regel L2:*Die vorher erstellte Einfugen-Transformation wird aus dem Abbildungsmodell gelöscht.

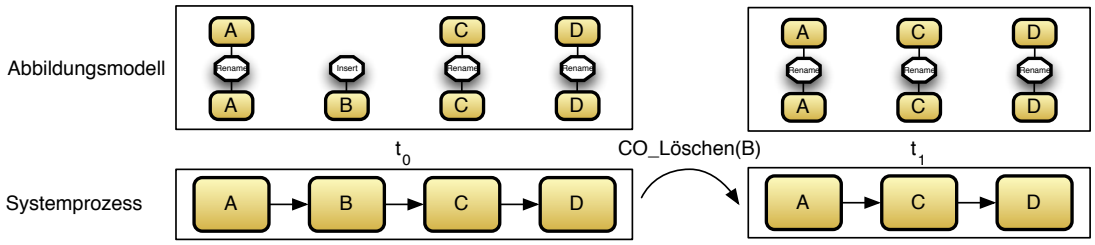

Abbildung 3.14: L2: Löschen Regel 2

*Regel L3:*Die Aufspalten-Transformation wird zu einer Umbenennen Transformation, da sie nur noch eine Systemtask besitzt.

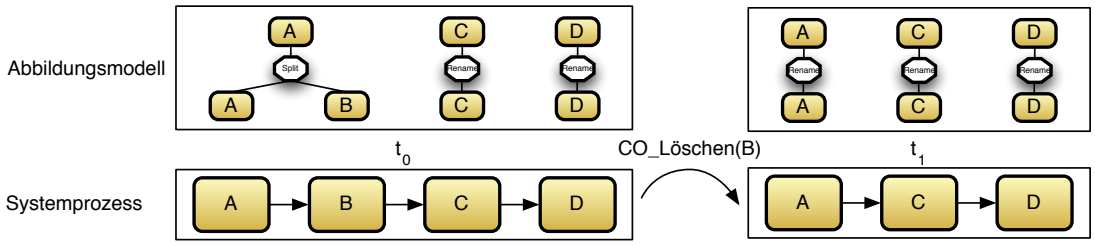

Abbildung 3.15: L3: Löschen Regel 3

*Regel L4:*Die Zusammenfassen-Transformation wird zu mehreren Einfüge-Transformationen, da keine Systemtasks mehr vorhanden sind.

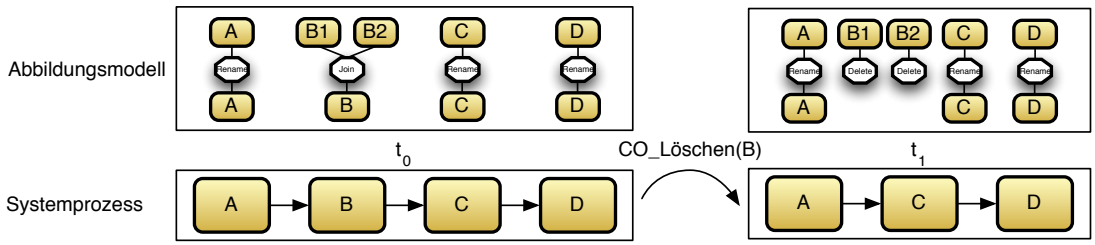

Abbildung 3.16: L4: Löschen Regel 4

Änderungsoperation(CO_Umbenennen): *Regel U1:*Eine Task A wird im Systemprozess umbenannt. Diese Bezeichnung muss im Abbildungsmodell ebenfalls geändert werden.

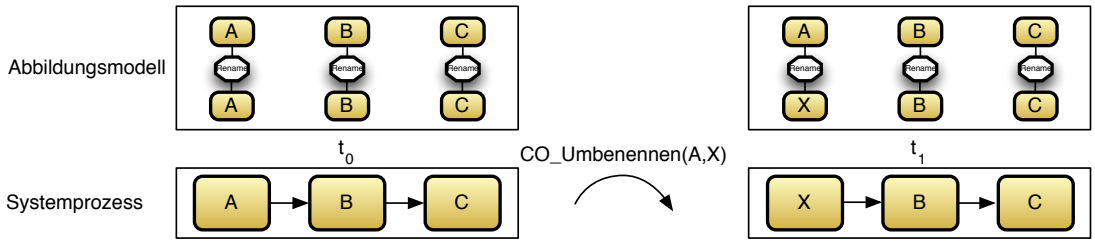

Abbildung 3.17: U1: Umbenennen Regel 1

Änderungsoperation(CO_Zusammenfassen): Zwei Tasks A und B werden im Systemprozess zusammengefasst. Abhängig davon, welchen Transformationen im Abbildungsmodell diese Tasks angehören, oder ob sich die Tasks in unterschiedlichen Transformationen befinden.

*Regel Z1:*Task A und B befinden sich jeweils in einer Umbenennen Transformation. Diese werden zu einer Zusammenfassen Transformation zusammengefasst.

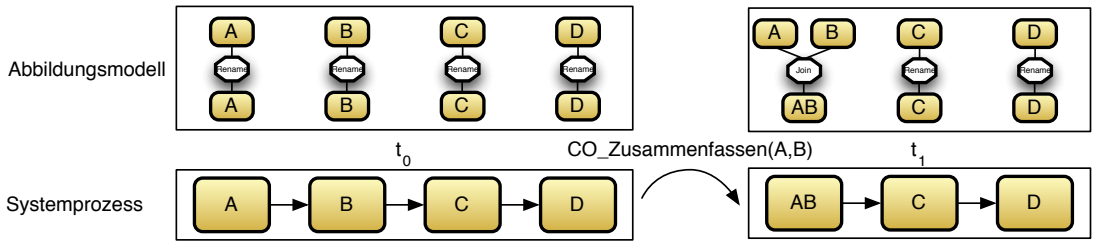

Abbildung 3.18: Z1: Zusammenfassen Regel 1

Regel Z2: Task A und B befinden sich jeweils in einer Einfüge-Transformation. Diese werden zu einer Einfüge-Transformation zusammengefasst.

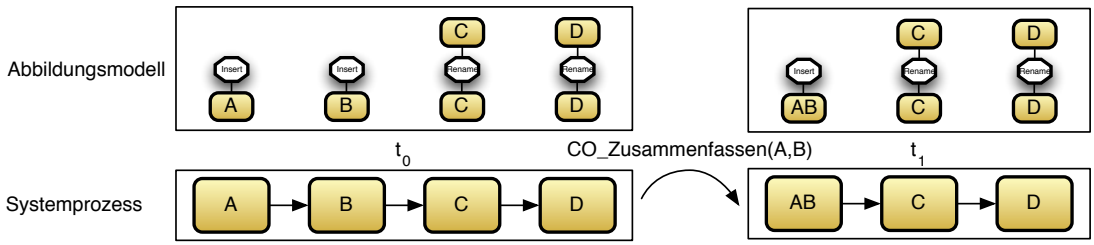

Abbildung 3.19: Z2: Zusammenfassen Regel 2

Regel Z3: Task A befindet sich in einer Umbenennen-Transformation, während B bereits zusammengefasst wurde. Es entsteht eine gemeinsame Zusammenfassen-Transformation.

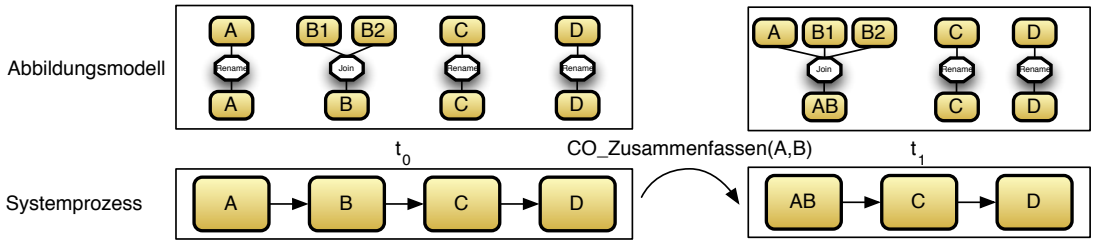

Abbildung 3.20: Z3: Zusammenfassen Regel 3

Regel Z4: Task A befindet sich in einer Umbenennen-Transformation, während B vorher eingefügt wurde. Diese Einfüge-Transformation verschwindet, und es entsteht eine gemeinsame Umbenennen-Transforamtion.

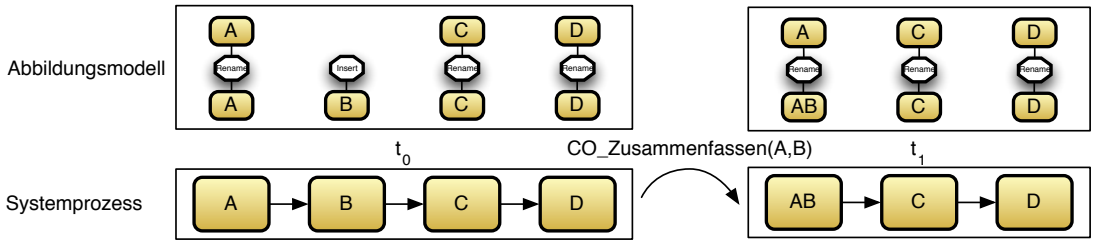

Abbildung 3.21: Z4: Zusammenfassen Regel 4

Regel Z5: Task A2 und B1 werden zusammengefasst und befinden sich jeweils in einer eigenen Aufspalten-Transformation. Hierbei entsteht eine Auswahl an alternativen Lösungsmöglichkeiten, bei denen die neue zusammengefasste Task A2B1 bei den verschiedenen beteiligten Transformationen nach den vorigen Regeln unterschiedlich verhalten kann.

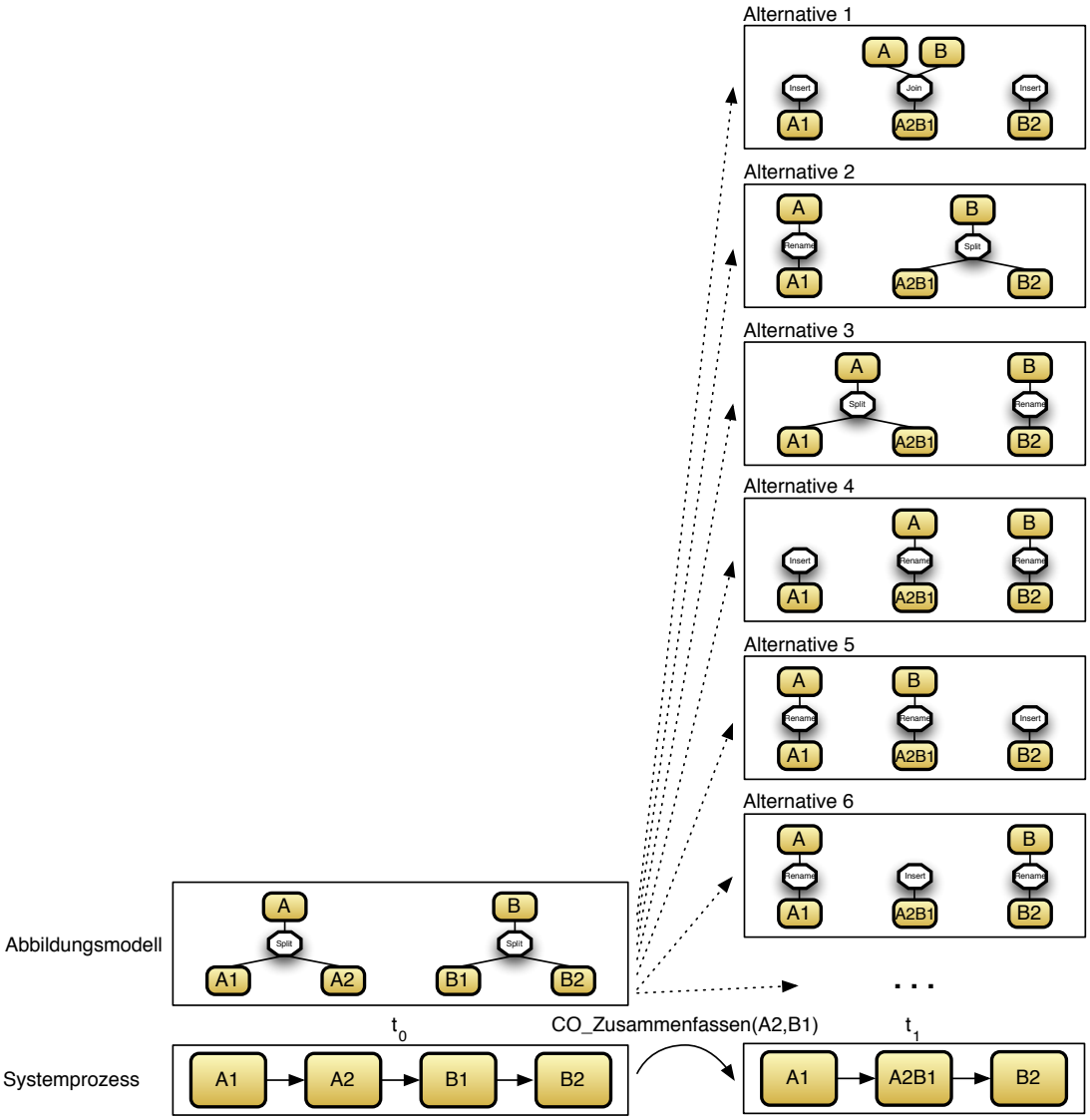

Abbildung 3.22: Z5: Zusammenfassen Regel 5

3.3.3 Fazit:

Methode 3 versucht aufbauend auf Methode 1 den Aufwand für den Systemmodellierer stetig zu verringern. Ein erster Schritt dafür, ist die automatische Konvertierung des Fachprozesses in den Systemprozess. Für die unterschiedlichsten Metasprachen existieren heutzutage Ansätze, um diese ineinander zu überführen, wodurch die Unterstützung bei diversen Modellierungswerkzeugen für diese Funktionen gleichzeitig wächst.
Der Systemmodellierer muss nun den Systemprozess nur noch teilweise abändern, um ihn dem ausführbaren Prozess anzupassen. Dies führt unter Umständen zu einem ersten Nachteil. Der Systemmodellierer kann in seinem verwendeten Werkzeug den Systemprozess nicht auf seine gewohnte Art und Weise frei Modellieren. Jede Änderung muss über die beschriebenen Änderungs-

operationen geschehen, welche zuallererst einmal durch das verwendete Werkzeug unterstützt werden müssen. Diese können jedoch auf verschiedenste Weise implementiert werden. In einer „guten“ Umsetzung können sich die Änderungsoperationen positiv auf den Erstellungsaufwand auswirken. Da die 5 Operationen, Aufspalten, Einfügen, Löschen, Umbenennen und Zusammenfassen eigens für die Umgestaltung des Fachprozesses in den Systemprozess definiert wurden, ist der Arbeitsablauf mit Hilfe der Änderungsoperationen für diesen Sachverhalt viel intuitiver.
Aus dem fertig umstrukturierten Systemprozess und dem Fachprozess wird nun automatisch das Abbildungsmodell generiert. Unabhängig von dem Zeitpunkt an dem das passiert, wird dem Bearbeiter dadurch fast der komplette Aufwand für die Erstellung des Abbildungsmodells abgenommen. Dies war ein großer Nachteil in Methode 1. Je größer die Prozesse werden, desto schwerer kann es sein bei abgeschlossener Umstrukturierung des Systemprozesses im Abbildungsmodell die richtigen Fachtasks auf die Systemtasks abzubilden. Viele Bezeichnungen haben sich in dieser Phase geändert, und mangels Übersicht können sich hier gravierende Fehler einschleichen.
Durch die Protokollierung der Änderungen, entweder in einer Tabelle, oder implizit über die sofortige Anwendung im Abbildungsmodell, die immer die richtigen Fachtasks mit den zusammengehörigen Systemtasks verbunden. Somit wird die Fehlerquelle der Erstellung des Abbildungsmodells minimiert.
Es kann jedoch nicht das gesamte Abbildungsmodell automatisch generiert werden. Wie bei den Transformationsregeln 3.11 und 3.22 ersichtlich, sind diese nicht immer eindeutig. Hier kann nicht generell entschieden werden, welche der Alternativen angewendet werden soll, wodurch der Bearbeiter an dieser Stelle eingreifen muss. Dies ist natürlich als Nachteil zu bewerten, wobei der Zeitaufwand für die Auflösung dieser Konflikte in einem erträglichen Rahmen sein sollte, was diesen Nachteil etwas abschwächt.

Im folgenden werden die Vor- und Nachteile der Methode 3 zusammenfassend aufgelistet:

+ Freie Modellierung des Systempozesses.
+ Bei Erstellung des Abbildungsmodells sind alle Tasks bereits definiert.
+ Automatisches Kopieren aller Tasks zwischen den Modellen.
+ Konsistenz zwischen allen 3 Modellebenen bei der Erstellung.
+ Automatische Generierung der Transformationsknoten zwischen Quell- und Zielobjekten des Abbildungsmodells.

Insgesamt ist diese Methode sehr positiv zu bewerten. Der Erstellungsaufwand für den Systemmodellmodellierer wurde gravierend gesenkt, und mögliche Fehlerquellen bei der Modellierung wurden im vergleich zu in Methode 1 abgeschwächt.

3.4 Methode 4 - Berechnung des Systemprozesses

In dieser Methode werden analog zu Methode 2 die Tasks des Fachprozesses in das Abbildungsmodell kopiert und anschließend daran der Systemprozess erstellt. Ziel dieser Methode ist es den Kontrollfluss des Systemprozesses so weit wie möglich vor zu generieren, um den Erstellungsaufwand für den Modellierer zu verringern. Hierzu sollen Kontrollflussinformationen aus dem Fachprozess im Abbildungsmodell gespeichert werden, um dadurch für den Systemprozess überführbare Fragmente zu identifizieren, welche zur Generierung verwendet werden können.

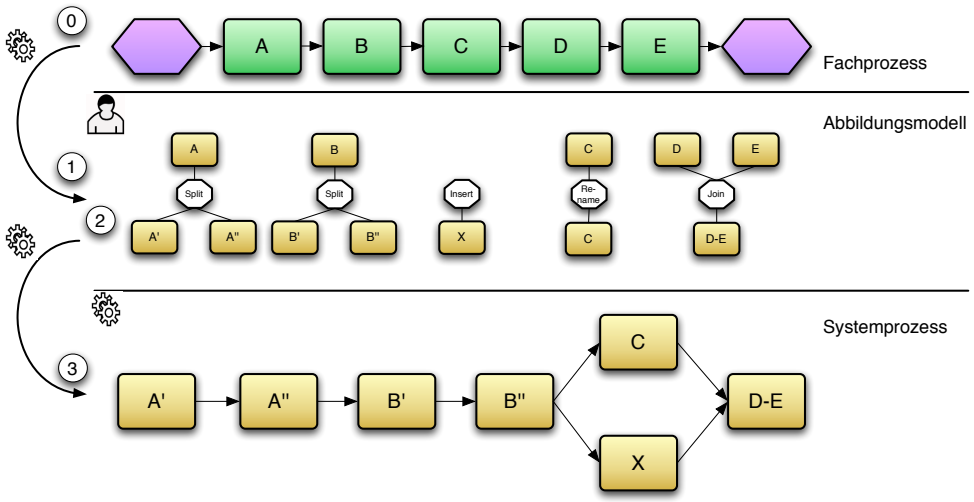

Abbildung 3.23: Übersicht Methode 4

3.4.1 Erstellung des Abbildungsmodells und des Systemprozesses

Modellierung des Fachprozesses ⓪: Analog zu den vorherigen Methoden wird auch hier ein modellierter Fachprozess vorausgesetzt.

Erstellung des Abbildungsmodells ①: Im ersten Schritt werden wie in Methode 2 alle Funktionen des Fachprozesses in das Abbildungsmodell kopiert, jedoch geschieht dieser Schritt hier automatisch. Wie in Methode 3 müssen die Fachtasks und Verzweigungen hierbei möglicherweise in ein anderes Metamodell konvertiert werden. Die neuen Elemente enthalten im Gegensatz zu ihren Pendants in Methode 3 alle Attribute aus dem Fachmodell, da davon ausgehend durch den Modellierer die Systemtasks erstellt werden sollen. Diese Attribute sind in Methode 3 nach dem ersten Kopiervorgang in den Systemprozess auch vorhanden, gehen jedoch bei der Umstrukturierung dieses Prozesses u.U. verloren.
An dieser Stelle werden sie absichtlich behalten, um den Systemmodellierer bei der Erstellung der Systemtasks zu unterstützen.

Erhaltung des Kontrollfluss im Abbildungsmodell: In dieser Methode ist die Hauptaufgabe des Abbildungsmodells die bestmögliche Speicherung des Kontrollflusses des Fachprozesses sowie

die entsprechende Anordnung der Fachtasks für die spätere Modellierung. Der Kontrollfluss soll im Systemprozess möglichst vollständig vorgeneriert werden, weshalb dieser zwischen den Quellobjekten bereits vollständig aus dem Fachprozess importiert werden muss. Dies ist vergleichbar mit Methode 3, wenn der Fachprozess in den Systemprozess kopiert wird. Jedoch kann in dieser Methode der Kontrollfluss im Abbildungsmodell nicht eingezeichnet werden, weshalb hier auf eine andere Lösung zurückgegriffen werden muss.
Beim Import der Fachtasks wird dafür zusätzlich zu den Objekten auch die Information über die jeweiligen Vorgänger(alternativ auch die Nachfolger) übertragen, wodurch der Kontrollfluss nachvollzogen werde kann. Die Speicherung kann einerseits in den Attributen der Tasks geschehen, oder alternativ durch das verwendete Werkzeug verwaltet werden.

Abbildung 3.24 zeigt, wie der Kontrollfluss durch Vorgänger- und Nachfolger Beziehungen in einer Liste gespeichert wird.

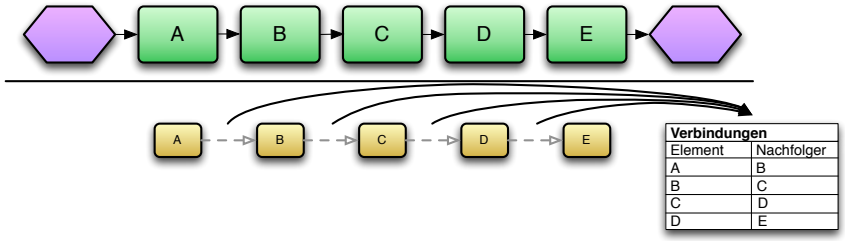

Abbildung 3.24: Verwaltung des Kontrollflusses im Abbildungsmodell

Ein weiterer Aspekt, der bei der Bearbeitung des Abbildungsmodells betrachtet werden muss, ist die übersichtliche Präsentation der einzelnen Transformationen. Die Modellierung der Systemprozesstasks findet hier innerhalb des Abbildungsmodells statt, weshalb eine ungünstige, dh. unzusammenhängende Anordnung der Tasks sich nachteilig für die Modellierung auswirken kann. Daher muss beim Import darauf geachtet werden zusammengehörige Tasks nebeneinander anzuordnen. Eine pragmatische Lösung ist es, im Fachprozess aufeinander folgende Tasks möglichst zusammenhängend im Abbildungsmodell anzuordnen, wobei dieser Ansatz bei Verzweigungen an seine Grenzen stößt. Dies ist jedoch für Fachprozesse mit wenigen alternativen Pfaden eine ausreichende Lösung.

Bearbeiten des Abbildungsmodells ②: Beim Bearbeiten des Abbildungsmodells geht der Modellierer analog zu Methode 2 vor. Jedoch werden in dieser Methode zur Vereinfachung der Modellierung für jede importierte Fachtask ein zugehöriger Systemtask mit Transformationsknoten vorgeneriert. Diese automatische Generierung geschieht an dieser Stelle wie in Methode 3 bei der Erstellung des Abbildungsmodells. Der Systemmodellierer ändert nun die erstellten Umbenennen-Transformationen dahingehend ab, dass diese den Fachprozess korrekt in den Systemprozess überführen.

Erstellung des Systemprozesses ③: Aus den modellierten Transformationen im Abbildungsmodell wird nun der Systemprozess generiert. Hierzu werden alle im Abbildungsmodell modellierten Tasks in den Systemprozess kopiert. Es wird nun versucht aus den Informationen, welche im

Abbildungsmodell vorhanden sind den Kontrollfluss zu generieren. Dafür werden die Anordnung der Fachtasks und Systemtasks berücksichtigt, sowie die kopierten Kontrollflussinformationen des Fachprozesses.

Generierung des Kontrollflusses im Systemprozess: Hierzu werden die Transformationsknoten mit zugehörigen Quell- und Zielobjekten des Abbildungsmodells nacheinander ausgewertet um die Systemtasks an der korrekten Stelle im Systemprozess zu erstellen.
Nachfolgend soll gezeigt werden, wie die unterschiedlichen Transformationstypen im Abbildungsmodell auf den Systemprozess abgebildet werden können, bzw. welche Probleme dabei auftauchen können.
diese müssen anschließend vom Systemmodellierer aufgelöst werden, da mehrere Auswahlmöglichkeiten existieren.

Einfügen: Die Task X wird im Abbildungsmodell über die Transformation-Einfügen zwischen den Zielobjekten C und D eingefügt (siehe Abbildung). Bei der Generierung des Systemprozesses muss nun entschieden werden, wo diese neue Systemtask eingefügt werden soll. Eine Möglichkeit ist es die „Nachbarschaft“ dieser Task im Abbildungsmodell zu betrachten. Abhängig davon, zwischen welchen Zielobjekten sich die Task befindet, wird sie im Systemprozess zwischen den zugehörigen Systemtasks eingefügt. Wie in Abbildung 3.25 weiter zu sehen ist, ist diese Regel nicht eindeutig anwendbar, wenn die Tasks C und D bereits im Fachprozess keine direkten Vorgänger bzw. Nachfolger waren. Es existieren nun zwei Möglichkeiten, wo diese Tasks eingefügt werden können.

1. Einfügen bei der ersten Aktivität
2. Einfügen bei der zweiten Aktivität

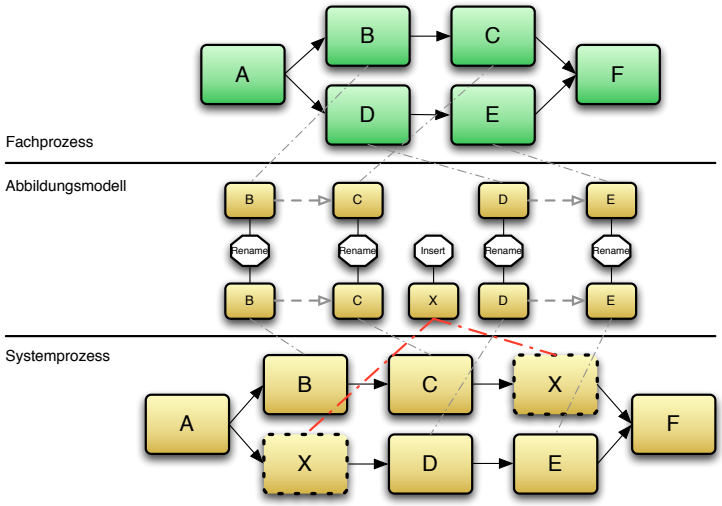

Abbildung 3.25: Einfügen einer neuen Task im Abbildungsmodell

Löschen: Die Quelltask B wird im Abbildungsmodell über Transformation-Löschen entfernt. Im Systemprozess werden der Vorgänger und Nachfolger miteinander verbunden.
Problematisch verhält sich diese Transformation, wenn eine Task im Fachprozess mehr als eine Eingangs- oder Ausgangskante besitzt. Dies ist bei der fachlichen Modellierung durchaus üblich, wenn in der Metamodellsprache Verzweigungen nicht explizit über eigene Strukturknoten modelliert werden müssen.

Abbildung 3.26 zeigt einen solchen Fall. Task B besitzt im Fachprozess zwei ausgehende Kanten und wird im Abbildungsmodell gelöscht. Im Systemprozess muss nun der Systemmodellierer entscheiden, wie dieser Konflikt aufgelöst werden soll.

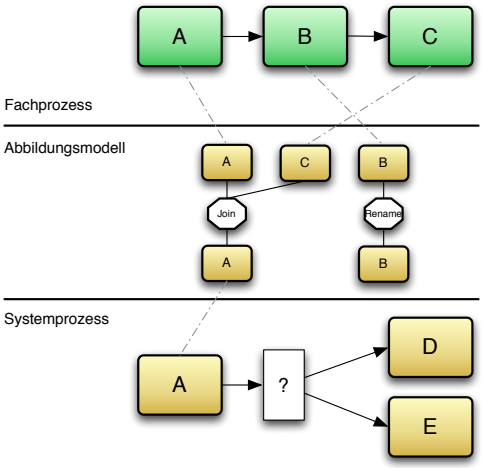

Abbildung 3.26: Löschen einer Task im Abbildungsmodell

Zusammenfassen: Mindestens zwei Quellobjekte werden im Abbildungsmodell vom Systemmodellierer über die Transformation-Zusammenfassen zu einem Zielobjekt zusammengefasst. Im Systemprozess muss nun die neu entstandene Systemtask an der richtigen Stelle eingefügt werden. Im einfachsten Fall sind diese Fachtasks direkte Nachfolger voneinander. Somit wird die neue Systemtask mit der Eingangskante der ersten zusammengefassten Task und mit der Ausgangskante der letzten verbunden.
Falls die Quellobjekte jedoch auf Fachtasks verweisen, die nicht mehr direkte Nachfolger voneinander sind, taucht die gleiche Problematik wie bei **Einfügen** auf. Es existieren für die zusammengefasste Task mehrere Stellen, an der sie im Systemprozess eingefügt werden kann.

Wie in Abbildung 3.27 zu sehen ist, werden die Quellobjekte, und somit Fachtasks B und D zu einer neuen Systemtask B-D zusammengefasst. Der Systemmodellierer muss bei der Generierung des Systemprozesses nun entscheiden, an welcher Stelle dieser eingefügt werden soll. Für diesen Fall existieren für die (teil-)automatisierte Erstellung des Systemprozesses 3 Möglichkeiten:

1. Zusammengefasste Task B an Stelle von Task B einfügen.
2. Zusammengefasste Task B an Stelle von Task D einfügen.
3. Task B in einen parallelen Zweig zu Task C einfügen.

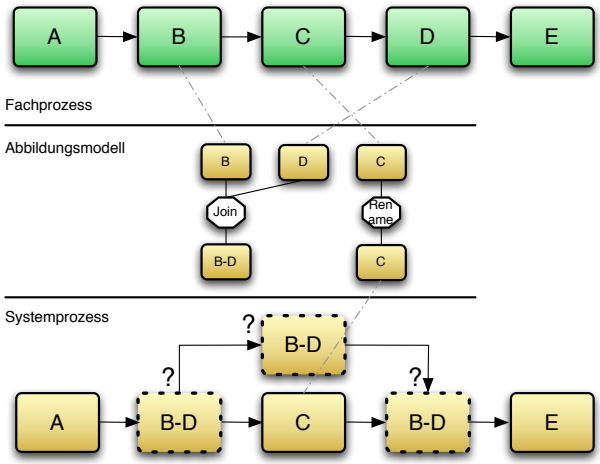

Abbildung 3.27: Zusammenfassen einer Task im Abbildungsmodell

Aufspalten: Ein Quellobjekt im Abbildungsmodell wird über die Transformation-Aufspalten in mehrere Zielobjekte aufgespalten. Im Systemprozess müssen diese Systemtasks nun an der richtigen Stelle in der richtigen Reihenfolge auftauchen. Für die automatische Generierung kann die Reihenfolge, in welcher die Quellobjekte im Abbildungsmodell angeordnet wurden, dafür verwendet werden die entsprechenden Systemprozesstasks in der gleichen Reihenfolge sequentiell anzuordnen.

Wie in Abbildung 3.28 zu sehen ist können die aufgespaltenen Systemtasks B1 und B2 jedoch auch parallel zueinander angeordnet werden. Dieser Fall ist wiederum problematisch, da im Abbildungsmodell keine Möglichkeit existiert herauszufinden, welche der beiden Anordnungen im Systemprozess vorgenommen werden soll.

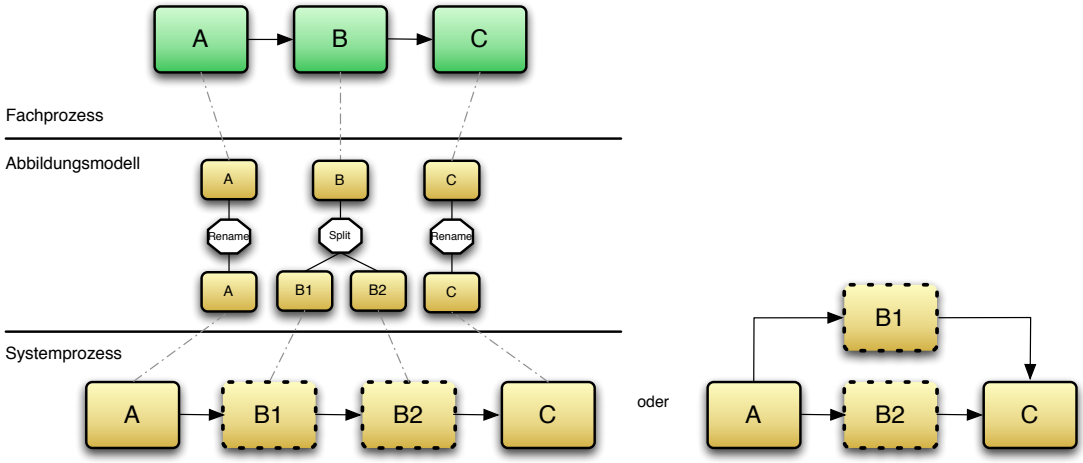

Abbildung 3.28: Zusammenfassen einer Task im Abbildungsmodell

3.4.2 Fazit

Methode 4 schwächt viele Nachteile der Methode 2 durch die (teil-)automatische Generierung einzelner Schritte ab. Zum einen fällt das manuelle Erstellen des Abbildungsmodells durch den Systemmodellierer weg. Wie in Methode 3 werden Abbildungsmodell und Systemprozess vorgeneriert, jedoch werden die Tasks des Fachprozesses zuerst in das Abbildungsmodell kopiert. Die Fachtasks werden hier eins zu eins importiert, wohingegen der Kontrollfluss nicht graphisch dargestellt werden kann. Die Speicherung geschieht entweder in den einzelnen Quellobjekten, indem die jeweiligen direkten Vorgänger bzw. Nachfolger in den Attributen gespeichert werden, oder durch eine Liste im Abbildungsmodell repräsentiert wird.
Für den Systemmodellierer vorteilhaft wird bei Erstellung des Abbildungsmodells wird für jedes Quellobjekt ein Transformationsknoten mit zugehörigem Zielobjekt angelegt, wodurch auf den vorliegenden Transformationen nur noch Änderungen erfolgen müssen. Dies nimmt dem Systemmodellierer einen Großteil des Erstellungsaufwands ab und ist hier vergleichbar mit Methode 3 bei Schritt ①. Für die Generierung des Systemprozesses in dieser Methode ist die Anordnung der im Abbildungsmodell erstellten Zielobjekte wichtig, da anhand dieser die neue Position der zugehörigen Systemtasks im Systemprozess bestimmt wird.
Dies ist nicht immer eindeutig möglich, weshalb dieser Schritt nur teilautomatisch ablaufen kann, und der Systemmodellierer im Zweifelsfall entscheiden muss. Durch dieses Eingreifen ist es möglich den Systemprozess vollständig aus dem Abbildungsmodell zu generieren. Für den Systemmodellierer bleiben bei der Modellierung jedoch einige Nachteile aus Methode 2 erhalten.
Der Erstellungsreihenfolge über das Abbildungsmodell bleibt weiterhin sehr unintuitiv. Der komplette Prozess wird zuerst über einzelne Transformationen entworfen, und erst bei Konflikten in Schritt ③ der Kontrollfluss für die neu entstandenen Systemtasks spezifiziert.

Zusammenfassend noch eine Übersicht über die Vor- und Nachteile dieser Methode:

+ Nur der Kontrollfluss muss im Systemprozess eingezeichnet werden.
+ Automatisches Kopieren aller Tasks zwischen den Modellen.
+ Vorgenerierung der Transformationsknoten zwischen Quell- und Zielobjekten des Abbildungsmodells.
+ Konsistenz zwischen allen 3 Modellebenen bei der Erstellung.
- Keine freie Modellierung des Systempozesses
- Definition der Systemtasks über das Abbildungsmodell.

Der Aufwand, welcher in dieser Methode für die Generierung des Systemprozesses betrieben wird steht im Endeffekt nicht in Relation zu dem Nutzen, d.h. Zeit- und Aufwandsersparnis, welcher dadurch erzielt werden soll. Daher fällt diese Methode in ihrer Gesamtheit hinter Methode 3 zurück, da diese den Systemmodellierer auf intuitivere Art effektiver unterstützt.

4

Konsistenzchecks und nachträgliche Änderung des Abbildungsmodells

Die Funktion des Systemmodells beschränkt sich nicht darauf, eine passive Zwischenschicht für Fachmodell und ausführbarem Modell zu bilden, sondern übernimmt bei nachträglichen Änderungen auch eine aktive Rolle.
Durch die Dokumentationen der Umstrukturierungen des Fachprozesses in den ausführbaren Prozess im Abbildungsmodell, können dort Änderungen bemerkt, und automatisch an die jeweils andere Modellebene weitergereicht werden.
Der jeweilige Bearbeiter hat nun die Möglichkeit, die in seinem Prozess durch Änderungen betroffenen Fach- oder Systemtasks anzeigen zu lassen, und entsprechend zu überprüfen.

4.1 Voraussetzungen

In Kapitel 3 wurden Methoden beschrieben, wie das Abbildungsmodell bei vorhandenem Fachprozess (und ggf. zu erstellenden Systemprozess) initial generiert werden kann. Während der Erstellung des Abbildungsmodells, schließen die Methodiken 3 und 4 Inkonsistenzen zwischen den Modellebenen aus. Das geschieht unter der Annahme, dass während der initialen Erstellung keine Änderungen am Fachprozess erfolgen.
Direkt nach der Erstellung sind Fachprozess, Abbildungsmodell und Systemprozess also untereinander Konsistent. Das bedeutet, dass das Abbildungsmodell alle Tasks aus Fach- und Systemprozess durch die Quell- und Zielknoten in ihrer aktuellen Version referenziert, und diese durch die korrekt zugeordneten Transformationsknoten aufeinander abbildet werden. Jedem Quell- und Zielknoten ist somit ein Transformationsknoten zugeordnet, und jedem Transformationsknoten ist die laut der Definitionen in Kapitel 2.5.4 erlaubte Anzahl an Quell- und Zielknoten zugeordnet. Daraus folgen für das Abbildungsmodell zwei Konsistenzkriterien, die unterschieden werden müssen. Zum einen die Konsistenz zwischen den Fachtasks und den Quellobjekten des Abbil-

dungsmodells sowie zwischen den Zielobjekten und den Systemtasks. Zum anderen die Konsistenz der Transformationen innerhalb des Abbildungsmodells.

4.2 Ursprung von Inkonsistenzen

Im Kontext eines Prozess-Lebenszyklus (siehe Abbildung 2.1) befinden sich die Prozesse nach der abgeschlossenen Entwurfs- und Implementationsphase im kontinuierlichen Betrieb. In dieser Phase können, entweder fachlich oder technisch motiviert, unvorhergesehene Prozessänderungen geschehen, welche die Konsistenz stören, und daher im folgenden ausführlich betrachtet werden sollen.

4.2.1 Fachliche Änderung

Im betrieblichen Umfeld muss auf Änderungen flexibel reagiert werden. Diese Änderungen sind beispielsweise eine kurzfristige Reaktion auf ein Konkurrenzprodukt, eine langwierige strategische Neuausrichtung des Unternehmens, oder eine sonstige Veränderung eines bestehenden Arbeitsablaufs. In jedem Fall ändert sich der geplante Geschäftsprozess, und somit das fachliche Modell. Es können beispielsweise Tasks hinzu kommen, entfernt, oder geändert werden.
Auf diese Änderungen muss im Systemprozess, und somit im ausführbaren Modell, reagiert werden, da diese fachlichen Änderungen meist eine technische Anpassung erforderlich machen. Für die Identifikation[1] der technischen Anpassungen bietet das Abbildungsmodell die Möglichkeit die Systemtask (und somit den Service) einer Fachtask zuzuordnen. Die Anpassung einer fachlichen Task kann auf der technischen Ebene auch mehrere verschiedene Services beeinflussen, welche beispielsweise von externen Service-Providern angeboten werden, weshalb eine schnelle Service Identifikation gewollt ist.

4.2.2 Systemprozessänderung

Auf der anderen Seite können auch im ausführbaren Modell Änderungen am bestehenden Prozess auftreten. Bei einer Service Abschaltung, beispielsweise von einem externen Service-Provider, muss im Systemmodell, und somit im Fachmodell darauf reagiert werden. Zuständige Fachmodellierer sollten auf die Problemstellung hingewiesen werden, um entsprechend zu reagieren. Eine mögliche Reaktion ist zu prüfen, welche Auswirkungen ein ausgefallener Service auf das Fachmodell hat.
Hier ist wie bei fachlichen Änderungen die Reaktionsgeschwindigkeit wiederum von hoher Bedeutung.

4.2.3 Problemstellung

Um auf diese Änderungen in den umliegenden Modellen des Abbildungsmodells zu reagieren, müssen Änderungen zunächst bemerkt werden. Da das Ergebnis einer fachlich, oder technisch

[1] Es wird die Identifikation eines bestehenden Services gemeint. Im Gegensatz dazu existiert im Umfeld von SOA noch der Begriff der Prozessidentifikation während der Analysephase.

getriebenen Änderung eine Differenz zwischen den Modellen ist, die noch nicht durch das Abbildungsmodell dokumentiert ist, ist es die Aufgabe von Konsistenzchecks, diese Differenzen über das Abbildungsmodell zu bemerken. Die weitere Vorgehensweise ist eine geeignete Anzeige, die Weiterleitung, und letztendlich die Auflösung durch den zuständigen Modellierer.
Das Überprüfen auf Differenzen und somit Konsistenz kann auf mehreren Wegen geschehen.
Eine Möglichkeit ist es Veränderungen im Fach- und Systemmodell zu überwachen und ereignisbasiert (bspw. beim Speichern) automatisch eine Prüfung auf Konsistenz durchführen. Weiter ist es denkbar diese Überprüfung auch in regelmäßigen Zeitabständen, oder bei Leerlaufzeiten zu beginnen.
Mögliche Differenzen, welche im Abbildungsmodell entdeckt werden können, sind unterschiedliche Taskmengen oder abweichende Timestamps. Eine geänderte Taskmenge deutet auf strukturelle Änderungen, wie beispielsweise das Einfügen oder Löschen einer Task hin, wohingegen verschiedene Timestamps auf eine Änderung einer bestehenden Task hinweisen. Die kann beispielsweise eine neue Version eines Services, oder ein geändertes Attribut einer Fachtask sein.

4.3 Methodik zur Auflösung von Inkonsistenzen

An dieser Stelle wird nun eine Methode vorgestellt, wie Inkonsistenzen mit Hilfe des Abbildungsmodells erkannt und beseitigt werden können. Dazu werden zunächst die weiter oben bereits erwähnten zwei Konsistenzkriterien formalisiert. Danach wird die Vorgehensweise erklärt und schlussendlich anhand des Anwendungsszenarios durchgeführt.

4.3.1 Konsistenzdefinitionen

Zur Durchführung eines Konsistenzchecks ist es nötig, die Kriterien zunächst zu formalisieren. Nachfolgend wird daher die Konsistenz innerhalb des Abbildungsmodells als *innere Konsistenz* bezeichnet, und die Konsistenz zwischen Fachmodell-Abbildungsmodell und Abbildungsmodell-Systemmodell als *äußere Konsistenz*.

Kriterien auf innere Konsistenz:

Definition 4.1
Das Abbildungsmodell ist nach ***innen konsistent****, wenn jeder Transformationstyp die korrekte Anzahl an Quell- und Zielknoten besitzt (siehe dazu Kapitel 2.5.4).*

Kriterien auf äußere Konsistenz:

Definition 4.2
Fachmodell und Abbildungsmodell sind konsistent, wenn folgende Regeln gelten:

- *Anzahl der Fachtasks = Anzahl der Quellobjekt*
- *Für jede Fachtask F existiert genau ein Quellobjekt Q mit* $F_{Name} = Q_{Name}$ *und* $F_{Timestamp} = Q_{Timestamp}$

Definition 4.3
Abbildungsmodell und Systemprozess sind konsistent, wenn folgende Regeln gelten:

- *Anzahl der Zielobjekt = Anzahl der Systemtasks*
- *Für jeden Zielobjekt Z existiert genau eine Systemtasks S mit* $Z_{Name} = S_{Name}$ *und* $Z_{Timestamp} = S_{Timestamp}$

Definition 4.4
Das Abbildungsmodell heißt nach ***außen konsistent****, wenn Fachmodell und Abbildungsmodell sowie Abbildungsmodell und Systemmodell konsistent sind.*

Mit Hilfe dieser Definitionen kann nun festgelegt werden, auf welche Weise Konsistenzchecks im Abbildungsmodell durchgeführt werden können.

4.3.2 Allgemeine Vorgehensweise

Abbildung 4.1 zeigt die allgemeine Vorgehensweise für die Überprüfung der Modellebenen auf Konsistenz.

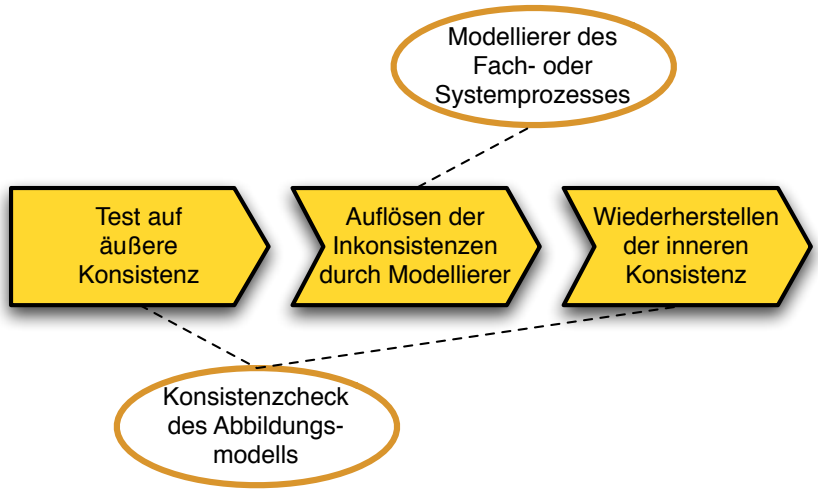

Abbildung 4.1: Vorgehen bei der Konsistenzüberprüfung

Um Änderungen im Fachmodell und Systemprozess zu bemerken wird zuerst auf äußere Konsistenz überprüft. Dadurch identifiziert das Abbildungsmodell Fach- und Systemtasks die fehlen, geändert, oder neu eingefügt wurden. Der Zuständige Fachmodellierer (oder Systemmodellierer) bekommt nun im Abbildungsmodell angezeigt, welche Tasks überprüft werden müssen, und passt das zuständige Modell bei Relevanz an. Im nächsten Schritt wird die innere Konsistenz im Abbildungsmodell hergestellt, indem das Abbildungsmodell die veränderten Transformationen entsprechend „repariert“.

Diese drei Schritte werden anhand des Anwendungsszenarios nun etwas genauer betrachtet. Für die Übersichtlichkeit wurden in Abbildung 4.2 wie in Kapitel 3 alle Bezeichnungen durch Buchstaben ersetzt.

```
Teste für jedes Quellobjekt: QuellObjekt == Fachtask
   Nein:    Fachtask gelöscht: markiere Quellobjekt als gelöscht
            Fachtask geändert: markiere Quellobjekt als geändert

            Finde zugehörige Zielobjekte und markiere diese zur Überprüfung
            durch den Systemmodellierer

Teste für jedes Zielobjekt: ZielObjekt == Systemtask
   Nein:    Systemtask gelöscht: markiere Zielobjekt als gelöscht
            Systemtask geändert: markiere Zielobjekt als geändert

            Finde zugehörige Quellobjekte und markiere diese zur Überprüfung
            durch den Fachmodellierer
```

Abbildung 4.3: Algorithmus 1: Test auf mögliche Inkonsistenzen

In dem Anwendungsszenario geschehen vier verschiedene fachliche Änderungen.

- Fachtask A wird durch den Fachbereich entfernt.
- Fachtask C wird durch den Fachbereich geändert.
- Fachtask E wird durch den Fachbereich gelöscht, besitzt jedoch keine entsprechende im Systemtask.
- Fachtask G wird durch den Fachbereich hinzugefügt.

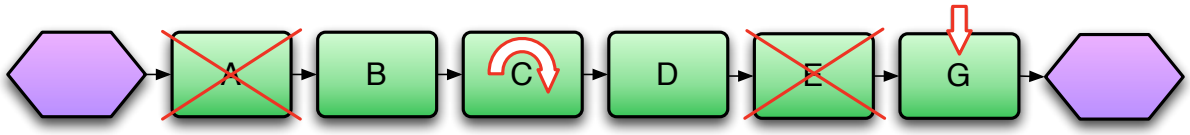

Abbildung 4.2: Fachliche Änderungen im Anwendungsszenario

Nach dem in Abbildung 4.1 festgelegten Vorgehen ist nun der erste Schritt auf äußere Konsistenz zu überprüfen.

4.3.3 Test auf äußere Konsistenz

Um auf äußere Konsistenz zu überprüfen, werden die in Definition 5.2 und 5.3 festgelegten Kriterien durch das Abbildungsmodell validiert. In einem Algorithmus sieht das folgendermaßen aus:

Dieser Algorithmus findet nun für das Anwendungsszenario vier Inkonsistenzen und markiert die entsprechenden Quell- und Zielobjekte. Dies wird in Abbildung 4.4 veranschaulicht.
In Inkonsistenz ① wird Quellobjekt A daher zum löschen markiert, und Zielobjekt A für den

Systemmodellierer zur Überprüfung durch ein ? markiert.
In Inkonsistenz ② wird Quellobjekt C als geändert markiert, und Zielobjekt C für den Systemmodellierer ebenfalls markiert.
Bei Inkonsistenz ③ bemerkt der Algorithmus die gelöschte Fachtask E, und markiert dementsprechend das Quellobjekt E als gelöscht. Da diese Transformation jedoch keine Zielobjekte besitzt, werden diese dem Systemmodellierer nicht angezeigt.
In der letzten Inkonsistenz ④ erstellt der Algorithmus für die neue Fachtask G ein neues Quell- und Zielobjekt, und markiert diese zur Überprüfung.

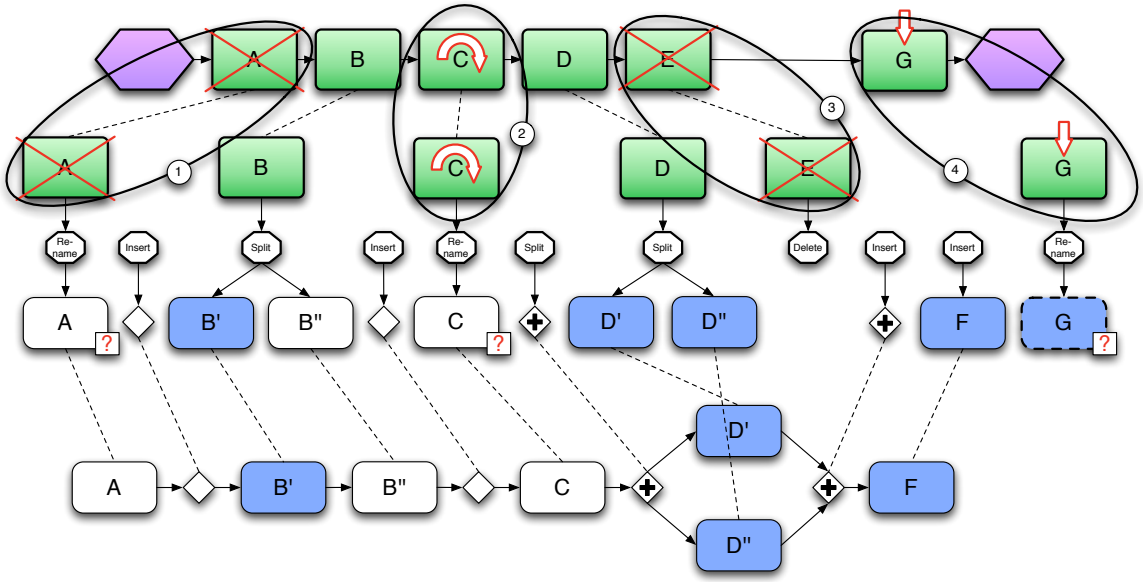

Abbildung 4.4: Markierungen durch Konsistenzcheck im Abbildungsmodell

4.3.4 Auflösen der Inkonsistenzen

Auf das Anwendungsbeispiel bezogen, ist nun der nächste Schritt, dass der Systemmodellierer für jedes markierte Zielobjekt entscheidet, was mit der zugehörigen Systemtask passiert, bzw. diese neu angelegt werden soll. Für die beiden markierten Zielobjekte A und C existieren nun zwei Möglichkeiten auf deren Inkonsistenz mit Systemtask A und C zu reagieren. Zum einen das Systemmodell so anzupassen, dass es die Systemtask nicht mehr enthält, oder zum anderen die Systemtask aus technischen Gründen zu behalten, obwohl die zugehörige Fachtask (wie es bei Systemtask A der Fall ist) gelöscht wurde.
Für das markierte Zielobjekt G kann der Systemmodellierer eine neue Systemtask erstellen, oder schließt die Änderung im Systemprozess ebenfalls aus technischen Gründen aus.

Der Systemmodellierer markiert nun im Abbildungsmodell diejenigen Zielobjekte, deren Systemtasks vollständig angepasst wurden. Die explizite manuelle Markierung ist für die Herstellung der inneren Konsistenz von Bedeutung, und wird in diesem Zusammenhang zu einem späteren Zeitpunkt erläutert.

Abbildung 4.5 zeigt die durch den Systemmodellierer abgearbeiteten Inkonsistenten. In diesem Szenario wurde Systemtask A (obwohl sie in der fachlichen Ebene gelöscht wurde) behalten. Systemtask C wurde ebenfalls nicht verändert, da sich die fachliche Änderung in diesem Fall nicht auf den Systemprozess auswirken soll. Systemtask G wurde jedoch im Systemprozess hinzugefügt. Die Markierung „bearbeitet“ für die Zielobjekte wird hierbei durch ein grünen Haken angezeigt.

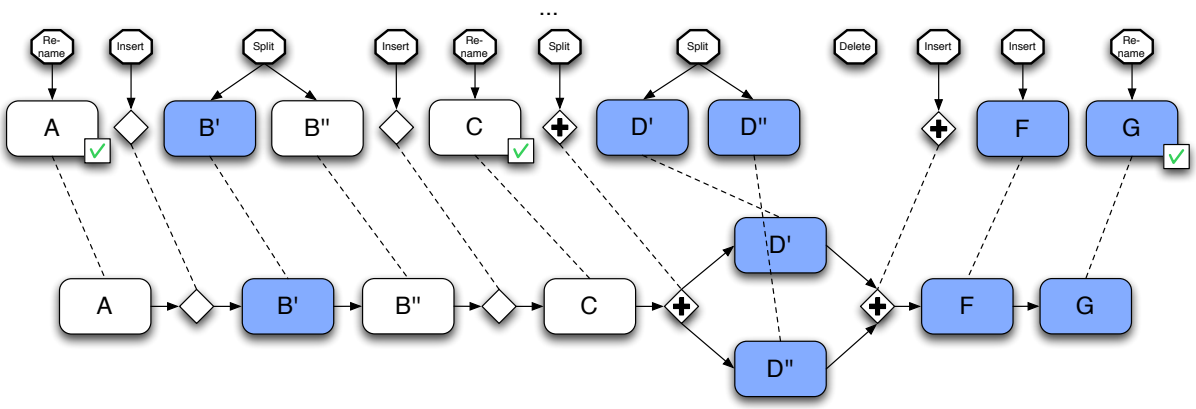

Abbildung 4.5: Markierungen durch Systemmodellierer im Abbildungsmodell

4.3.5 Herstellen der inneren Konsistenz

Aufgabe des Abbildungsmodells ist es nun, die innere Konsistenz herzustellen. Dieser Schritt besteht aus zwei Teilen. Zuerst werden im Beispiel alle Quellobjekte gelöscht, die durch die äußeren Konsistenzkriterien „zum Löschen“ markiert wurden. Danach wird durch die für jedes markierte Zielobjekt erkannt, ob es eine zugehörige Systemtask besitzt. Ist dies nicht der Fall, wird das Zielobjekt ebenfalls gelöscht. Dieser Automatismus ist der Grund für die manuelle Markierung abgearbeiteter Zielobjekte durch den Systemmodellierer.
Letztendlich müssen noch die Typen für alle beteiligten Transformationsknoten dahingehend angepasst werden, dass die in Definition 5.1 festgelegte Bedingung für innere Konsistenz erfüllt wird. Folgender Algorithmus führt diesen Schritt durch:

```
Für jedes markierte Quellobjekt:
   Löschen:    Lösche das Quellobjekt
   Geändert:   Ändere das Quellobjekt:
               Quellobjekt = Fachtask

   Vom Bearbeiter Markiert: Fachtask vorhanden?
      Ja:     Quellobjekt = Fachtask
      Nein: Lösche Quellobjekt

Für jedes markierte Zielobjekt:
   Löschen:    Lösche das Zielobjekt
   Geändert:   Ändere das Zielobjekt:
               Zielobjekt = Systemtask

   Vom Bearbeiter Markiert: Systemtask vorhanden?
      Ja:     Zielobjekt = Systemtask
      Nein:   Lösche Zielobjekt

Für jeden Transformationsknoten:
   Ändere Typ des Knotens, sodass Definition 5.1. erfüllt ist
```

Abbildung 4.6: Algorithmus 2: Wiederherstellen der inneren Konsistenz

Im Anschluss daran befinden sich die Modelle untereinander wieder in einem konsistenten Zustand.

4.3.6 Schlussbemerkungen

Der Algorithmus zur Herstellung der Konsistenz wurde in dem Anwendungsszenario nur ausgehend von Fachmodelländerungen betrachtet. Dies geschah rein aus Überlegungen die Methodik darzustellen und die Reaktion der Konsistenzchecks auf Änderungen in den verschiedenen Ebenen vorzustellen. Die vorgeschlagenen Algorithmen 1 und 2 sind darauf ausgelegt auch zeitgleiche Änderungen zumindest zu bemerken. Bei konkurrierenden Änderungen von Fachmodell und Sy-

stemprozess auf die selbe Transformation kann die Auflösung der Inkonsistenzen nicht mehr allein durch Fach- oder IT-Bereich geschehen. Dieses Problem leitet sich daraus ab, dass hier ein konkurrierender Zugriff auf das Abbildungsmodell auftritt, und Veränderungen verloren gehen könnten. Hier existiert noch Untersuchungsbedarf für eine geeignete Methode.

5 Prototyp

In diesem Kapitel wird die in Kapitel 3 vorgestellte Methode 3 für die Erstellung des Abbildungsmodells in einem Prototypen umgesetzt. Dafür wird das ARIS Business Architect verwendet. Hiebei handelt es sich um ein Werkzeug zur Modellierung von Geschäftsprozessen sowie zur Prozessanalyse. Die Entscheidung ARIS zu benutzen, beruht auf der Möglichkeit, das Programm weitgehend anzupassen. Dafür bietet ARIS eine auf JavaScript basierende Scriptsprache „ArisScript“, die es erlaubt Ereignisse im Editor abzufangen sowie Objekte und Modelle in der Datenbank zu erstellen, zu löschen und zu bearbeiten. Die Software an sich ermöglicht (für die prototypische Umsetzung besonders wichtig) die Definition eigener Objekt- und Modelltypen.

5.1 Einführung in ARIS Business Architect

Wie bereits in der Einführung zu diesem Kapitel erwähnt wurde, verwaltet ARIS Business Architect alle Objekte und Modelle in einer eigenen Datenbank. Auf diese hat man durch eigens definierte Filter die Möglichkeit, nicht gewollte Elemente für die Bearbeitung auszublenden. In dem Prototyp wurde für die Implementierung daher ein eigener Filter *Diplomarbeit* definiert, der in den ebenfalls separat definierten Modelltypen Fachprozess, Abbildungsmodell und Systemprozess nur relevante Elemente anzeigt. Abbildung 5.1 zeigt eine Übersicht über die eigens definierten Modelltypen sowie über die für diesen Prototyp durch einen Filter zugelassenen Objekte.

Das Konzept von ARIS beschreibt einerseits eine Menge von Datenbankobjekten, und andererseits deren zugeordnete Ansichten. Diese Datenbankobjekte werden *Objektdefinitionen* genannt. Jede Sicht auf diese Objektdefinition wird durch eine sog. *Objektausprägung* im Editor angezeigt. Abbildung 5.2 zeigt den Zusammenhang zwischen diesen Elementen.

Die Motivation für die Aufspaltung von Objekten in Definition und Ausprägung in der Modellstruktur des ARIS-Konzeptes (siehe ARIS-Haus bei 2.4.1), ist die Notwendigkeit gleiche Objekt

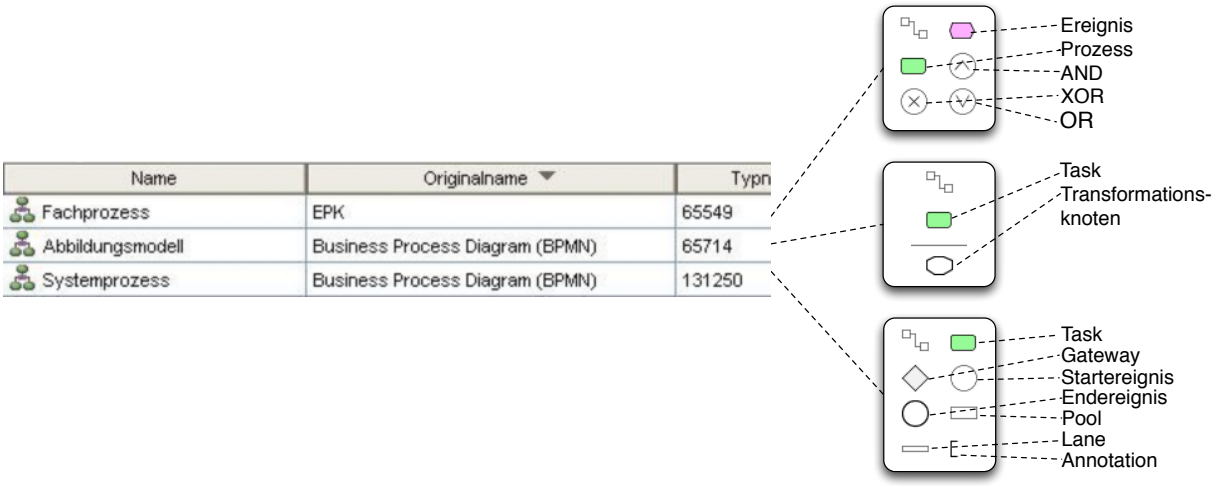

Abbildung 5.1: Filter Diplomarbeit

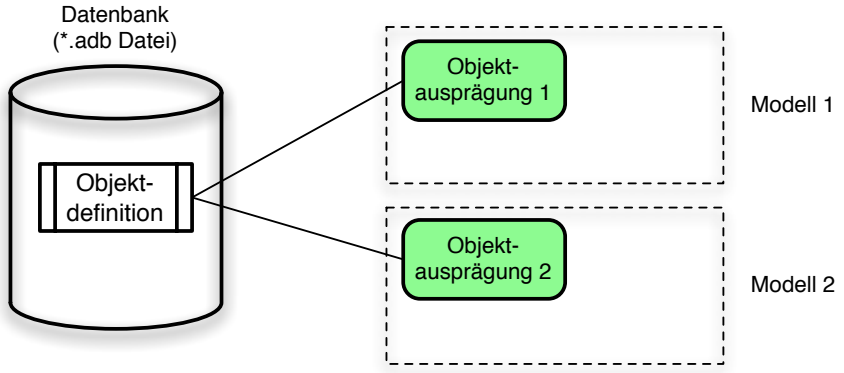

Abbildung 5.2: Objektdefinitionen und Objektausprägungen

in unterschiedlichen Modellen anzuzeigen. Um die Beziehung zwischen diesen mehrfach vorhandenen Objekten zu behalten und somit die Navigation unter den Modellen zu verbessern, wurde diese Aufteilung vorgenommen.

Der nun vorgestellte Prototyp besteht in ARIS im wesentlichen aus zwei Teilen. Zum einen aus den neuen Modellen (Fachmodell, Abbildungsmodell, Systemprozess) die sich aus bestehenden Typen ableiten sowie aus den neu erstellten Objekten wie bspw. den Transformationsknoten. Zum anderen aus Scripten, die die Funktionalität für die Modellerstellung und Überprüfung der Konsistenz implementieren.

5.2 Übersicht über die Implementierung

Die bereits eingeführten Modellebenen(Fachmodell, Abbildungsmodell, Systemprozess) werden im Prototyp gemeinsam in ARIS implementiert. Dies führt zu keiner Verletzung der Verallgemeinerung der Methode auf andere Werkzeuge, da in diesem Fall einerseits unterschiedliche Modelltypen verwendet werden, und andererseits zu keinem Zeitpunkt von einem Report eine Gesamtsicht auf alle Modellebenen existiert. Dadurch lässt sich das Vorhandensein der Werkzeuggrenze zwischen Fach- und Systemmodell dahingehend simulieren, dass im Abbildungsmodell nur Referenzen auf die Objekt-ID's der Tasks im Systemprozess, als auch im Fachprozess existieren. Das bildet einen realen Anwendungsfall dahingehend ab, dass dies eine Untermenge der Informationen darstellt, die bei einem Export des Fach- und Systemprozesses in ein anderes Werkzeug importiert werden. Dem Abbildungsmodell ist weder bekannt, in welcher Reihenfolge die Tasks in dem jeweiligen Prozess angeordnet sind, noch in welcher Metasprache sie dort modelliert wurden.

Aus dem Grund, dass das Abbildungsmodell aus Sicht der Modellebenen zum Systemmodell gehört, wird in diesem Prototyp zwischen diesen beiden Modellen jedoch derselbe Modelltyp verwendet.

Die für den Prototyp verwendete Modellstruktur wird in dem folgenden Schaubild dargestellt.

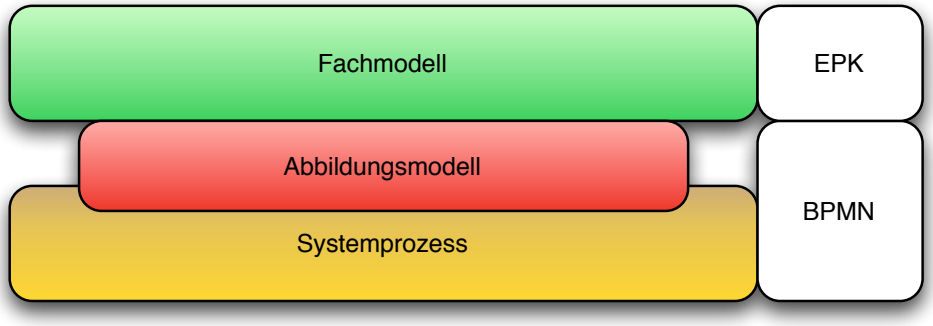

Abbildung 5.3: Modelltypen im Prototyp

Im Fachprozess werden sowohl für die Implementierung in ARIS, als auch für die Veranschaulichung der fachlichen Modellierung Ereignis Prozess Ketten (siehe 2.4.1) verwendet. Für den Systemprozess, und somit auch für das Abbildungsmodell werden BPD's (siehe 2.4.3) erstellt. Die Beschreibung des Systemprozesses in BPMN ist dahingehend Sinnvoll, dass der ausführbare Prozess, sofern er in einer anderen Notation als BPMN vorliegt, voll durch diese spezifiziert werden kann [GX09, ODH, YSWS08, LVD08]. Die Entscheidung das Abbildungsmodell auch in BPMN zu beschreiben wurde wegen der „Nähe“ zum Systemprozess bereits begründet.

Wie bereits erwähnt besitzt das Abbildungsmodell für jeder seiner Tasks eine Referenz auf die zugehörige Task im Fach- und Systemprozess. Aufgrund von technischen Gegebenheiten in ARIS, welche zu einem späteren Zeitpunkt erläutert werden, müssen für diese Implementierung auch im Systemprozess Referenzen auf die zugehörigen Tasks im Abbildungsmodell gehalten werden. Wie dies in einer alternativen Implementierung umgangen werden kann wird an dieser Stelle ebenfalls erläutert.

In ARIS besitzt jedes Objekt eine eindeutige globale Identifikationsnummer, die sogenannte GUID. Diese Globale Objekt-ID erlaubt es ein Objekt, in diesem Fall ein Element des jeweiligen Prozesses, innerhalb einer Datenbank eindeutig zu referenzieren. Dies geschieht über ein benutzerdefiniertes Attribute, welches bei Erstellung einer Task im Abbildungsmodell gesetzt wird und auf das Ursprungselement im Fach- oder Systemprozess verweist.

Die folgende Abbildung 5.4 enthält eine Übersicht über die Referenzierung der Objekte über die drei Modelle.

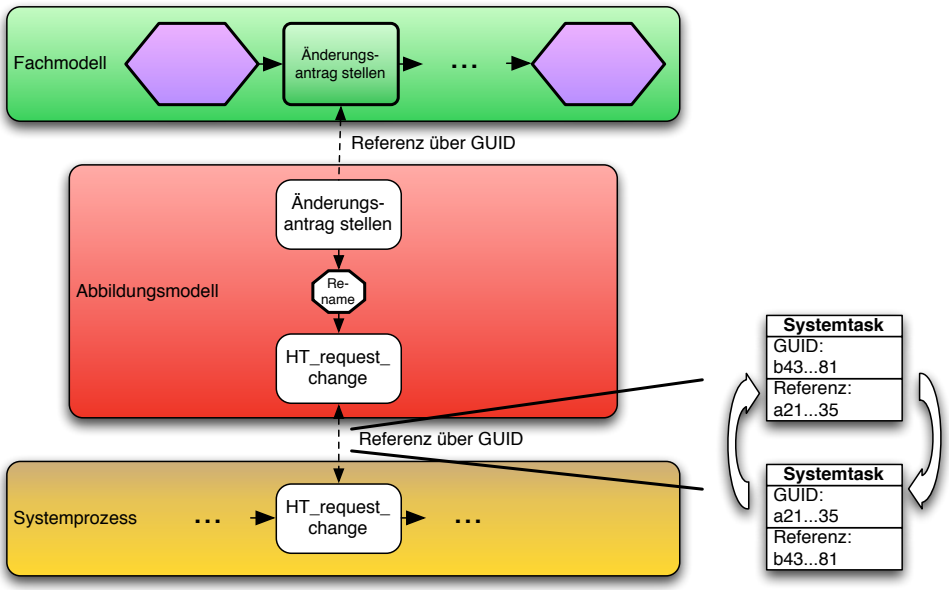

Abbildung 5.4: Herstellen der Referenzen über die Modellebenen

Zwischen den Modellebenen existieren demnach manuell erstellte Verbindungen zwischen Objekten durch Referenzen auf die GUID, wohingegen im Abbildungsmodell diese Beziehungen explizit über Pfeile und Transformationknoten modelliert werden.

Ein weiteres Detail zur prototypischen Umsetzung ist die Festlegung, welche Elemente aus Fach und Systemprozess aufeinander abgebildet werden sollen.
Da der Fachprozess in EPK vorliegt, und das Abbildungsmodell eine Abbildung dieses Modells auf den Systemprozess (BPMN) vornimmt, muss vorher geklärt werden, welche Elemente aufeinander abgebildet werden (vgl. dazu Abbildung 5.5). Die Überführung eines EPK-Diagramms in ein BPMN-Diagramm beschränkt sich in dem Prototyp auf die wichtigsten Elemente, da die Umsetzung des Abbildungsmodells und dessen Generierung im Vordergrund steht.
Die einzelnen Überführungen werden noch einmal bei der Transformation des Fachprozesses in den Systemprozess in Absatz 5.3.1 besprochen, daher wird dies an dieser Stelle zurückgestellt.

Das Abbildungsmodell wird zusätzlich um die bereits eingeführten Transformationsknoten erweitert. ARIS erlaubt es, eigene Objekt- und Symboltypen aus bestehenden abzuleiten. Der Transformationsknoten wird in dem Prototypen aus einem Gateway abgeleitet. Ein Transformationsknoten soll auf der einen Seite mehrere Tasks zusammenfassen können, als auch mehrere Tasks aufspalten können, weshalb mehrere eingehende bzw. ausgehende Kanten für den Trans-

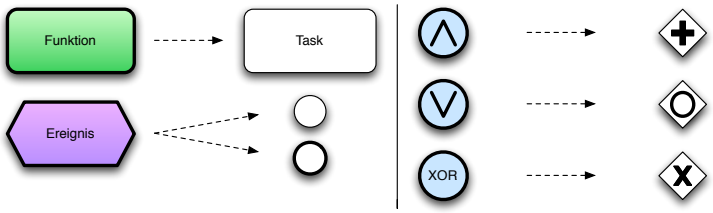

Abbildung 5.5: Abbildung von EPK auf BPMN im Prototyp

formationsknoten bei der Modellierung durch den Editor „erlaubt“ sein müssen. Im folgenden werden die Einzelheiten der prototypischen Implementierung vorgestellt. Dafür wird die Methodik 3 aus Kapitel 3 implementiert. Die Entscheidung diese Methode zu verwenden, beruht darauf, dass diese die meisten Vorteile in Bezug auf intuitive Erstellung und praktischer Anwendbarkeit mitbringt (siehe dazu das Fazit zu Methode 3 und 4). Ausßerdem sind die benötigten Änderungsoperationen für den Systemprozess in ARIS durch die eigens definierbaren Makros sehr gut umsetzbar.

5.3 Initiales Erstellen der Modelle

Dieser Abschnitt befasst sich nun mit der konkreten Erstellung des Prototypen. Der erste Schritt ist es nun, wie in Methode 3 beschrieben, den Fachprozess zu erstellen. Dieser kann in dem Editor von ARIS wie gewohnt in EPK modelliert werden. Abbildung 5.6 zeigt den Fachprozess des Anwendungsbeispiels aus Kapitel 1 in ARIS.

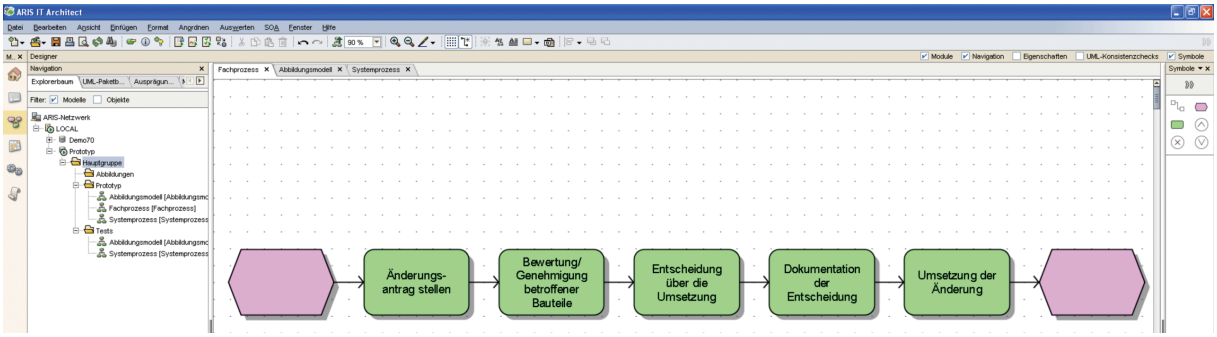

Abbildung 5.6: Fachprozess in ARIS modelliert

Innerhalb des Fachprozesses hat der Modellierer die Möglichkeit über ein Makro die Erstellung des Systemprozesses und somit auch des Abbildungsmodells zu starten. Dieser Aufruf benötigt keine weiteren Informationen, da die Konvertierung des Fachprozesses in den Systemprozess automatisch erfolgt. Der durch das Makro aufgerufene Report erstellt, gleichzeitig zu dem Systemprozess, das Abbildungsmodell, damit während der Erstellungsphase die beiden Modelle bei jeder Änderung konsistent gehalten werden können. Abbildung 5.7 zeigt anhand eines Ausschnitts der Übersicht aus Kapitel 3.3, welcher Schritt nun durch den Report durchgeführt wurde.

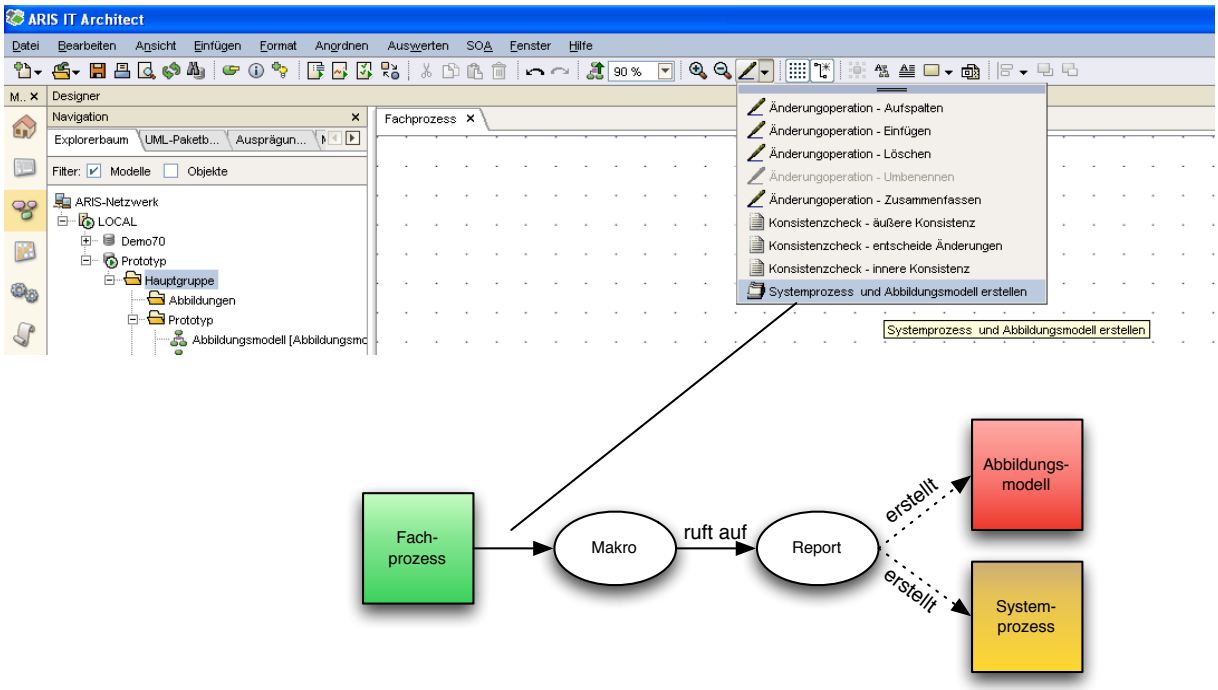

Abbildung 5.7: Erstellen von Systemprozess und Abbildungsmodell

5.3.1 Erstellung des Systemprozess

Für die Erstellung des Systemprozesses wird zunächst ein neues Modell angelegt, und die relevanten Tasks des Fachprozesses kopiert. Dies erfolgt über eine Kopie der Objektdefinition.
Da für den Prototyp eine Modelltransformation von EPK in BPMN stattfindet, werden nicht alle Elemente, wie beispielsweise einige Ereignisse übernommen.
In EPK beginnt und endet jeder Prozess mit einem Ereignis. Diese werden auf ein entsprechendes Start-, bzw. Endereignis in BPMN abgebildet. Funktionen in EPK lassen sich eins zu eins auf die Aktivitäten eines BPMN Diagramms abbilden, daher existieren in der Hinsicht keine Problemstellen.
Für Verzweigungen in EPK existieren ebenfalls semantisch gleichbedeutende Verzweigungen in BPMN.

Der Kontrollfluss wird zeitgleich mit der Objektkopie erstellt. ARIS bietet für zwei zu verbindende Objekte eine Auswahl innerhalb des aktuellen Filters erlaubter Verbindungstypen an (siehe Abbildung 5.8). Dadurch kann es rein syntaktisch zu keinen Problemen kommen, da immer die korrekten Verbindungstypen erstellt werden.

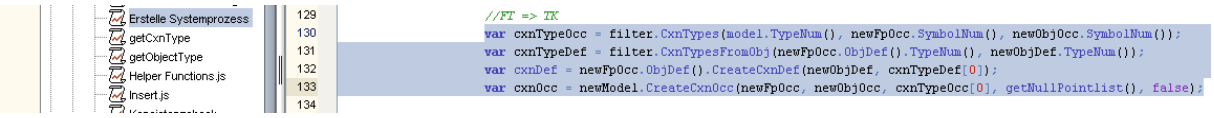

Abbildung 5.8: Anzeigen zulässiger Verbindungstypen unter ARIS

5.3.2 Erstellung des Abbildungsmodells

Da das Abbildungsmodell zeitgleich mit dem Systemprozess erstellt wird, wird im Gegensatz zu der allgemeinen Vorgehensweise in Kapitel 3.3, diese Erstellung vor den Änderungsoperationen auf dem Systemprozess beschrieben.

Für jedes Element, welches im Systemprozess neu generiert wird, wird im Abbildungsmodell eine eigene Objektdefinition erstellt. Für den Prototyp wäre es möglich, hier auch eine Ausprägungskopie zu erstellen, welche auf dasselbe Datenobjekt zugreift (siehe Abbildung 5.2). Der Vorteil wäre es, dass sich Änderungen auf der Task im Systemprozess (Name, Attribute, Änderungsdatum) implizit auf die Ausprägung im Abbildungsmodell auswirken. Hierbei würde jedoch die klare Trennung zwischen den Modellebenen verwischen, und Konsistenzuntersuchungen bei späteren Fach- und Systemprozessänderungen erschwert. Das resultiert daraus, dass die Konsistenzchecks die Fachtasks und Quellobjekte sowie Zielobjekte und Systemtasks miteinander verglichen werden (siehe Kapitel 2.5.4), dazu jedoch später mehr.
Durch das Erstellen eines eigenen Objektes muss, wie eingangs schon erwähnt, die Referenz auf das zugehörige Objekt im Systemprozess manuell gepflegt werden. Dafür wird ein „benutzerdefiniertes Attribut“ erstellt, welches die GUID der Systemtask enthält.

Der folgende Code zeigt einen Auszug aus dem Report, welcher die Erstellung des Systemprozesses und des Abbildungsmodells durchführt:

```
1  function ErstelleModelle(fachprozess)
2      {
3      var systemprozess = ErstelleSystemprozess(fachprozess);
4      var abbildungsmodell = ErstelleAbbildungsmodell();
5
6      for each(fpElement in fachprozess)
7          {
8          var spElement = ErstelleObjektInSystemprozess(fpElement);
9          var abbElement = ErstelleObjektInAbbildungsmodell(spElement);
10         }
11     }
```

Die dadurch generierten Modelle sind in Abbildung 5.9 zu sehen. Der Systemprozess wurde aus dem Fachprozess generiert, und die einzelnen Elemente aus EPK in ihre BPMN Equivalente überführt. Das Abildungsmodell ist daher zu diesem Zeitpunkt nur mit entsprechenden „Umbenennen“-Transformationen gefüllt.

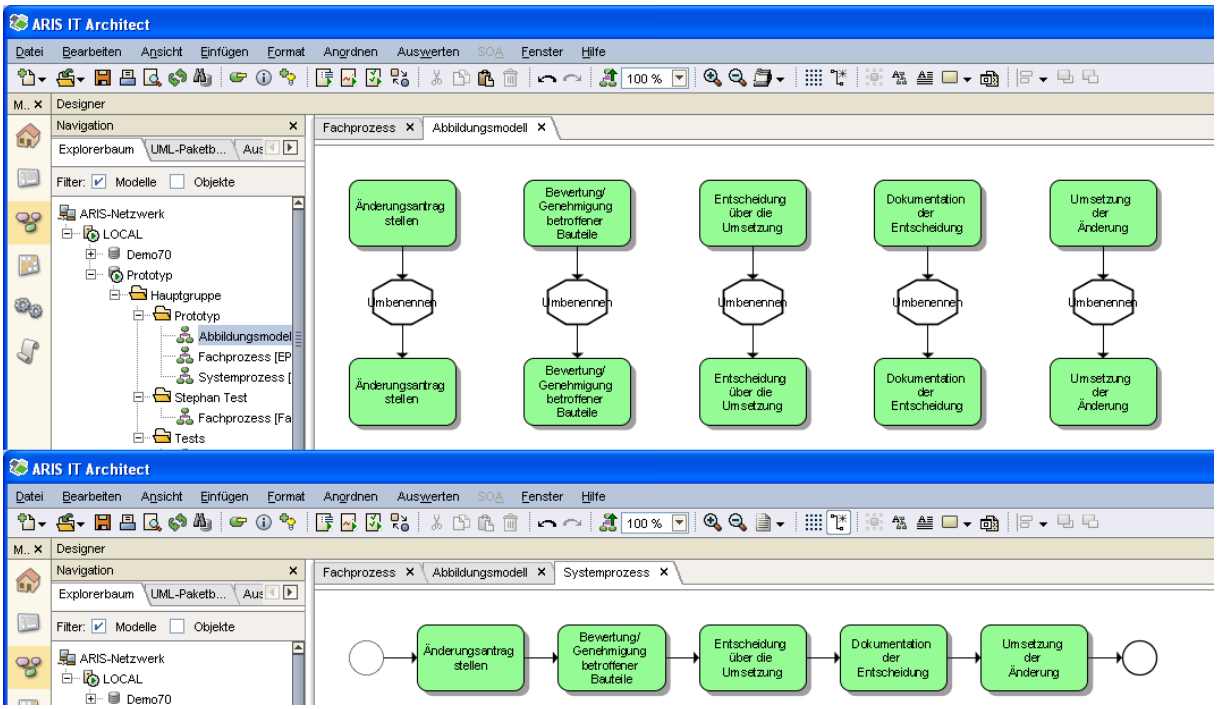

Abbildung 5.9: Initial erstellte Modelle

5.4 Änderungen im Systemprozess

Der vollständig in den Systemprozess überführte Fachprozess wird nun vom Implementierer sukzessive abgeändert. Dafür wird für jede mögliche Änderung an der Objektmenge des Systemprozesses eine eigene Änderungsoperation implementiert. Dies ist dahingehend nötig, dass dadurch jeder Änderung im Abbildungsmodell eindeutig die richtige Transformation zugeordnet werden kann (siehe dazu die dafür definierten Regeln bei 3.3.2).

Die Änderungsoperationen sind wiederum als Reports implementiert, welche über Makros komfortabel vom Implementierer aufgerufen werden können (siehe Abbildung 5.10). Das aufgerufene Makro übergibt für jede Änderungsoperation die eventuell ausgewählten Objekte an den Report, welche dann dazu benutzt werden die zugehörigen Transformationen im Abbildungsmodell zu identifizieren. Dies geschieht durch die am Anfang des Kapitels erwähnten Referenzen innerhalb der Systemtasks auf die Zielobjekte.
Eine Referenz auf das Abbildungsmodell wird dem Systemprozess der Einfachheit halber als benutzerdefiniertes Attribut hinterlegt.

Jede Änderungsoperation verhält sich im Abbildungsmodell je nach vorliegenden Transformationen unterschiedlich, weshalb nachfolgend die genaue Funktion der Änderungsoperationen dokumentiert wird. Die folgenden Operationen stellen den Aufruf der implementierten Reports dar, die für jede Änderungsoperation gesondert umgesetzt werden.

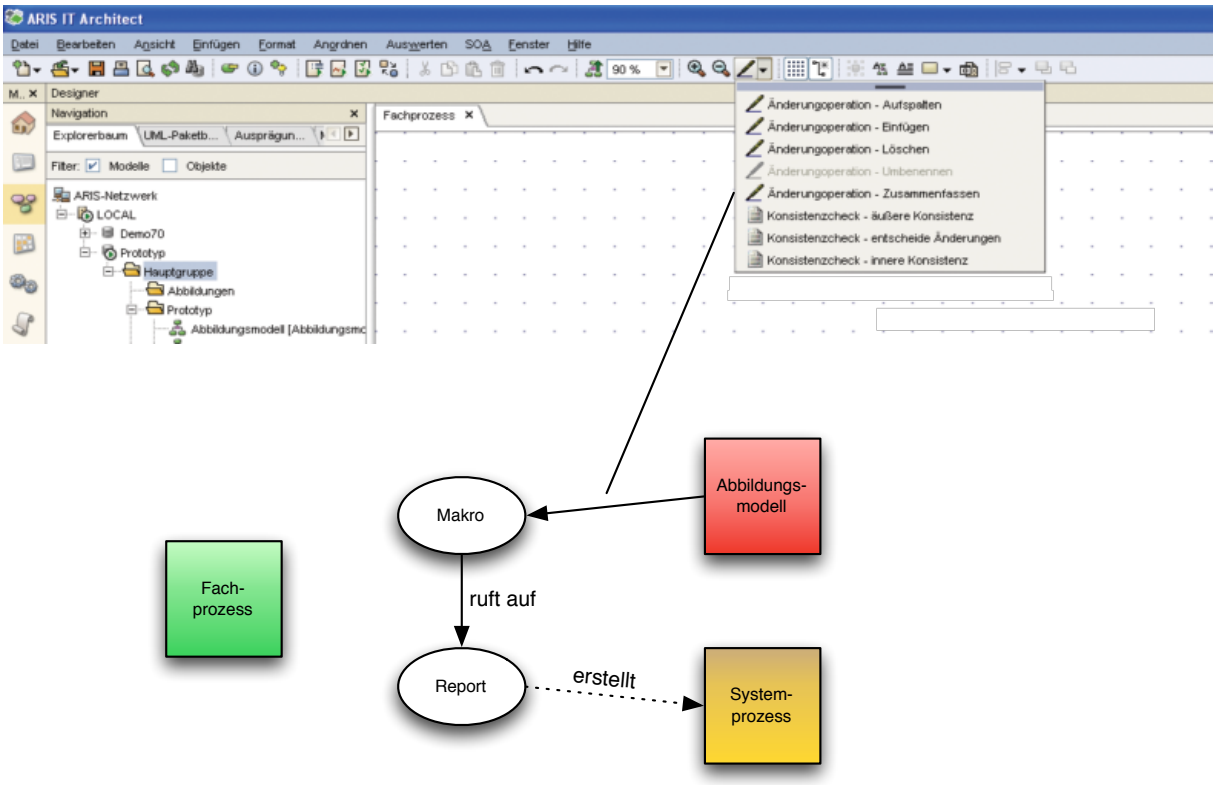

Abbildung 5.10: Aufruf von Änderungsoperationen im Systemprozess

Aufspalten: Eingabeparameter: ObjOcc SystemprozessTask, <String>Array NeueTaskNamen

Der Implementierer markiert im Systemprozess die Task die aufgespalten werden soll, und ruft das Makro(bzw. den Report) auf, welches im Abbildungsmodell die zugehörigen Änderungen vornimmt. Dafür werden die neuen Namen der aufgespaltenen Tasks benötigt, welche über einen Dialog vom Bearbeiter abgefragt werden. Der Report sucht nun im Abbildungsmodell den zugehörigen Transformationsknoten, und das zugehörige Quell- bzw. Zielobjekt. Wie in Methode 3 unter 3.3.2 dokumentiert, wird mit den eben identifizierten Elementen verfahren.

```
01 function Aufspalten(spTask){
02 var abbildungsmodell = getAbbildungsmodell();
03 //Zielobjekt im Abbildungsmodell holen
04 var abbTask = FindTaskInAbbildungsmodell(spTask);
05 var neueTasks = AufspaltenDialog(); //neue Tasknamen abfragen
06 var transformationsknoten = FindTransformationsknoten(abbTask);
07 //Abhängig von dem vorhandenen TK
08 //Ausnahmefälle abarbeiten und neue Objekte anlegen
09 if(transformationsknoten.ObjDef().Name(nLocId).equals(„Einfügen“))
10     {
11     AufspaltenNachEinfügen(abbTask, neueTasks);
12     var transDef = transformationsknoten.ObjDef();
```

```
13      transformationsknoten.Remove(true);
14      group.Delete(transDef);
15      }
16  else if(transformationsknoten.ObjDef().Name(nLocId).equals(,,Aufspalten‘‘))
17      {
18      AufspaltenNachAufspalten(abbTask, neueTasks, transformationsknoten);
19      }
20  else if(transformationsknoten.ObjDef().Name(nLocId).equals(,,Zusammenfassen‘‘))
21      {
22      AufspaltenNachZusammenfassen(abbTask, neueTasks, transformationsknoten);
23      var transDef = transformationsknoten.ObjDef();
24      transformationsknoten.Remove(true);
25      group.Delete(transDef);
26      }
27  else if(transformationsknoten.ObjDef().Name(nLocId).equals(,,Umbenennen‘‘))
28      {
29      AufspaltenNachUmbenennen(abbTask, neueTasks, transformationsknoten);
30      var transDef = transformationsknoten.ObjDef();
31      transformationsknoten.Remove(true);
32      group.Delete(transDef);
33      }
34  //Entfernen der ersetzten Tasks
35  DeleteOldTasks(spTask, abbTask);
36  //Layout im Abbildungsmodell zurücksetzen
37  MakeLayout();
38  }
```

Einfügen: Eingabeparameter: ObjOcc SystemprozessTask

Für das Einfügen wird beim Erstellen der neuen Task, oder des neuen Strukturknotens im Systemprozess ereignisbasiert ein Makro ausgeführt, welches die neue Task mit zugehörigem Transformationsknoten im Abbildungsmodell einfügt. Dazu wird im Abbildungsmodell für den Zielknoten eine neue Objektdefinition mit entsprechender Ausprägung erstellt. Außerdem kann hier sofort ein zugehöriger Einfügen-Transformationsknoten erstellt, und mit dem Zielobjekt verbunden werden. Hierfür existieren keinerlei Spezialfälle, da keine Interaktionen mit bestehenden Transformationsknoten stattfinden.

```
01  function Einfügen(spTask){
02  var abbildungsmodell = getAbbildungsmodell();
03  //Neue Task im Abbildungsmodell
04  var abbTask = CreateNewTaskInAbbildungsmodell(spTask);
05  var CreateNewTransformationsknoten(abbTask);
06  //Layout im Abbildungsmodell zurücksetzen
07  MakeLayout();
08  }
```

Zusammenfassen: Eingabep.: <ObjOcc>Array SystemprozessTasks, String NeuerTaskName

Der Bearbeiter markiert mehrere Tasks im Systemprozess und führt das Makro zum zusammenfassen dieser Tasks manuell aus. Der dadurch aufgerufene Report identifiziert die betroffenen Transformationen im Abbildungsmodell, und führt alle verbundenen Quellobjekte zu dem neuen, durch einen Dialog festgelegten, Zielobjekt zusammen. Hier wird, wie in Methode 3 beschrieben, dem zusammengefassten Task Vorrang gewährt. Das bedeutet, dass sonstige mit den früheren Transformationsknoten verbundene Zielobjekte, welche nicht zusammengefasst werden, nur noch über Einfügen-Transformationen im Abbildungsmodell abgebildet werden.
Der Algorithmus entfernt dafür alle zusammengefassten Tasks im Systemprozess und Abbildungsmodell, und erstellt dafür neue Tasks unter dem neuen Namen in beiden Modellen.

```
01 function Zusammenfassen(spTasks, newTask){
02 abbildungsmodell = getAbbildungsmodell();
03 var abbTasks = FindTasksInAbbildungsmodell(spTasks);
04 var transformationsknoten = FindTransformationsknoten(abbTasks);
05 var fachtasks = FindQuellObjekte(transformationsknoten);
06 var übrigeTasks = FindAllTasks(transformationsknoten, abbTasks);
07 Zusammenfassen(fachtasks, neueTask);
08 InsertOldTasks(übrigeTasks);
09 DeleteTransformationsknoten(transformationsknoten);
10 DeleteOldTasks(spTasks, abbTasks);
11 //Layout im Abbildungsmodell zurücksetzen
12 MakeLayout();
13 }
```

Umbenennen: Eingabeparameter: ObjOcc SystemprozessTask

Eine Systemtask wird im Editor im Systemprozess umbenannt. Dies wird wie beim Löschen und Umbenennen durch ein Makro ereignisbasiert abgefangen und der zugehörige Report ausgeführt. Die Operation identifiziert zu dem übergebenen Systemtask das entsprechende Zielobjekt im Abbildungsmodell, und ändert den Namen dementsprechend ab. Hierbei kann es zu keinen Fehlern bei der Ausführung kommen, da keine anderen Transformationen betroffen sind.

```
01 function Umbenennen(spTask){
02 var abbildungsmodell = getAbbildungsmodell();
03 var newName = Dialogs.InputBox(,,Wie ist der neue Name der Task?‘‘, ,,Umbenennen‘‘,
04 ,,newName‘‘);
05 var abbTask = FindTaskInAbbildungsmodell(spTask);
06 writeNodeName(abbTask, newName);
07 writeNodeName(abbTask, newName);
08 //Layout im Abbildungsmodell zurücksetzen
09 MakeLayout();
10 }
```

Löschen: Eingabeparameter: ObjOcc SystemprozessTask

Beim Löschen wählt der Implementierer im Systemprozess eine Task aus, und löscht diese. Dieses Ereignis wird abgefangen, und führt automatisch ein Makro aus, welches durch einen Report im Abbildungsmodell das zugehörige Zielobjekt entfernt. Die betroffene Transformation muss gegebenenfalls angepasst werden, wenn dadurch zu wenige Zielobjekte verbunden sind.

```
01 function Löschen(spTask){
02 var abbildungsmodell = getAbbildungsmodell();
03 var abbTask = FindTaskInAbbildungsmodell(spTask);
04 var transformationsknoten = FindTransformationsknoten(abbTask);
05 //Abhängig von dem vorhandenen TK
06 //Ausnahmefälle abarbeiten und neue Objekte anlegen
07 if(transformationsknoten.OutEdges(Constants.EDGES_ALL).length == 1)
08     {
09     writeNodeName(transformationsknoten, „Löschen“);
10     }
11 else if(transformationsknoten.OutEdges(Constants.EDGES_ALL).length == 2)
12     {
13     writeNodeName(transformationsknoten, „Umbenennen“);
14     }
15 //Tasks im Abbildungsmodell löschen
16 if(!group.Delete(abbTask.ObjDef()) || !group.Delete(spTask.ObjDef()))
17     {
18     //Dialogs.MsgBox("Fehler beim Löschen der Objektdefinitionen!");
19     }
20 //DatenObjekte löschen
21 abbTask.Remove(true);
22 spTask.Remove(true);
23 //Layout im Abbildungsmodell zurücksetzen
24 MakeLayout();
25 }
```

Nach Anwendung der eben beschriebenen Operationen befindet sich das Abbildungsmodell im konsistenten Zustand zu Fach- und Systemprozess (siehe Abbildung 5.11). Das liegt daran, dass jede Änderung am Systemprozess sofort im Abbildungsmodell angewendet wurde.
Jede Entscheidung, die bei einer Änderungsoperation getroffen werden muss, wie beispielsweise die Festlegung des Typs der zu erstellenden Transformationsknoten (siehe dazu Kapitel 3.3.2), wird sofort getätigt.
Nach der Erstellung und der Bearbeitung aller Modelle, ist nun die nächste Aufgabe für den Prototyp auf nachträgliche Änderungen in Fach- und Systemprozess zu reagieren. Dies geschieht mit Hilfe der im vorigen Kapitel eingeführten Konsistenzchecks.

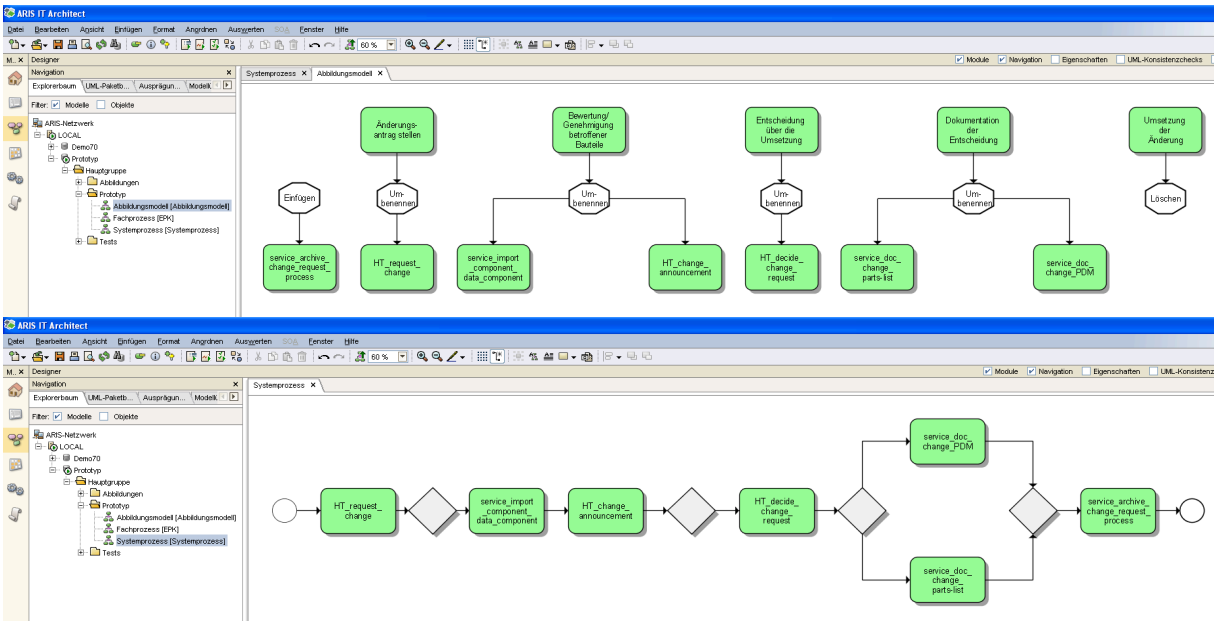

Abbildung 5.11: Verändertes Abbildungsmodell und veränderter Systemprozess

5.5 Konsistenzchecks durch den Prototyp

Die Methodik zur Erkennung von Konsistenzkonflikten wurde, wie in Kapitel 4 beschrieben, umgesetzt. Die allgemeine Vorgehensweise ist daher an die Reihenfolge aus diesem Kapitel angelehnt. Dazu wird, wie in Abbildung 5.12 gezeigt, durch einen Report zuerst auf äußere Konsistenz geprüft, und im Anschluss daran im Editor für das Abbildungsmodell alle Konflikte angezeigt. Danach werden die Konflikte manuell aufgelöst und abschließend wird wiederum durch einen Report die innere Konsistenz hergestellt.

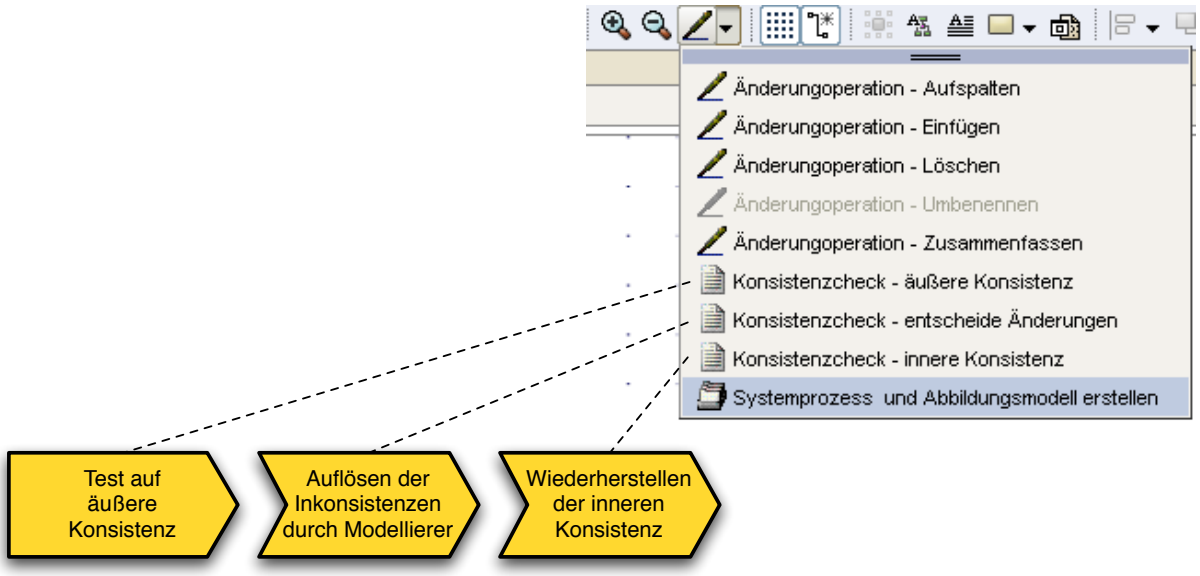

Abbildung 5.12: Übersicht über Konsistenzchecks im Prototyp

Um im Abbildungsmodell die Konflikte nach dem Test auf äußere Konsistenz darzustellen, müssen betroffene Transformationen und die zugehörigen Quell- und Zielobjekte durch den Report markiert werden. Dazu wird die Hintergrundfarbe der Quell- und Zielobjekte geändert (in diesem Fall gelb).

Im folgenden werden die einzelnen Schritte zur Überprüfung auf Konsistenz genauer betrachtet.

5.5.1 Äußere Konsistenz

Voraussetzung für die Überprüfung der äußeren Konsistenz ist die komplette Erstellung des Abbildungsmodells und des Systemprozesses.

Die Überprüfung auf die äußere Konsistenz erfolgt, wie für die Änderungsoperationen (sowie beim initialen Erstellen des Systemprozesses), durch das manuelle aufrufen eines Reports in der Symbolleiste. Dazu wird ein geöffnetes Modell benötigt, in dessen Kontext der Report aufgerufen wird. Die zugehörigen Modelle werden dann durch den Report automatisch ermittelt.

Für die Demonstration der implementierten Konsistenzchecks, wurde der Fachprozess wie in Kapitel 3 modelliert (siehe Abbildung 5.13), und eine Task eingefügt, ein Task entfernt sowie 2 Tasks geupdated.

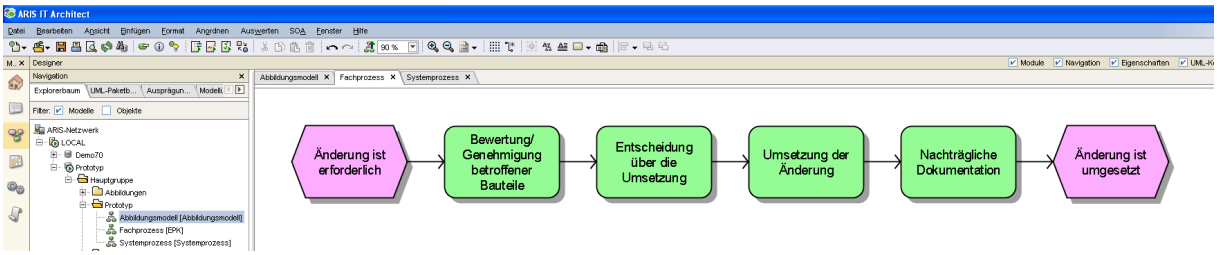

Abbildung 5.13: Der veränderte Fachprozess in ARIS

Anhand dieser Änderungen im Fachprozesses wird die Überprüfung auf äußere Konsistenz gestartet. Der Report ermittelt automatisch den zum Abbildungsmodell zugehörigen Fachprozess und Systemprozess, um mögliche Inkonsistenzen in den Modellen zu finden.

Diese werden anhand des in 4.3.3 vorgestellten Algorithmus 1 ermittelt. Die konkrete Implementierung verfährt wie in diesem Algorithmus beschrieben, und markiert die entsprechenden Transformationen sowie Quell- und Zielobjekte entsprechend (siehe Abbildung 5.14).

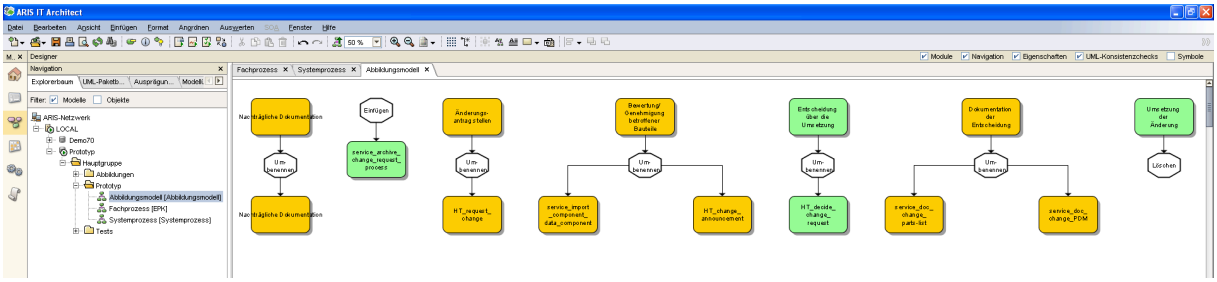

Abbildung 5.14: Das markierte Abbildungsmodell

Für neu hinzu gekommene Quell- und Zielknoten wird im jeweiligen Fachmodell oder Systemprozess direkt eine neue Task und ein Transformationsknoten erstellt ①. Dies geschieht bereits an dieser Stelle, damit einerseits die Beziehungen zwischen Quellobjekt und Fachtask sowie zwischen Zielobjekt und Systemtask in die Attribute geschrieben werden kann (Referenzen über GUID's). Andererseits kann der Modellierer diese Tasks bei Bedarf einfacher in das jeweilige Modell einbinden.
Die übrigen Inkonsistenzen ② ③, genauer die Differenz der Taskmengen zwischen Fach- und Systemprozess, werden im Abbildungsmodell ebenso erkannt und gelb markiert.

Die relevante Quellcode dieses Reports ist nachfolgend aufgelistet und dokumentiert.

```
01 function ÄussereKonsistenz()
02     {
03     var abbildungsmodell = getAbbildungsmodell();
04
05     korrekt = new Array(); //Korrekt modellierte Quellobjekte
```

```
06      neuImFp = new Array(); //Fachtasks, die nicht im Abb.mod. vorhanden
07      neuImAbb = new Array();//Quellobjekte die nicht im Fachprozess
08      quellobjekte = new Array(); //Alle Quellobjekte
09      updatedFp = new Array(); //Alle geänderten Quellobjekte
10
11      korrektSp = new Array(); //Korrekt modellierte Zielobjekte
12      neuImSp = new Array(); //Systemtasks, die nicht im Abb.mod. vorhanden
13      neuImAbbSp = new Array();//Zielobjekte, die nicht im Systemprozess
14      zielobjekte = new Array(); //Alle Zielobjekte
15      updatedSp = new Array(); //Alle geänderten Zielobjekte
16
17      FillArrays(korrekt, neuImFp, neuImAbb, quellobjekte, updatedFp,
18          korrektSp, neuImSp, neuImAbbSp, zielobbjekte, updatedSp);
19
20      for(var i=0; i<updatedFp.length; i++)
21          {
22          var ziele = FindZielobjekte(FindTransformationsknoten(updatedFp[i]));
23          neueMarkierungen.push(updatedFp[i]);
24          }
25          for(var i=0; i<neuImAbb.length; i++)
26          {
27          var ziele = FindZielobjekte(FindTransformationsknoten(neuImAbb[i]));
28          neueMarkierungen.push(neuImAbb[i]);
29          }
30      for(var i=0; i<updatedSp.length; i++)
31          {
32          var ziele = FindQuellobjekte(FindTransformationsknoten(updatedSp[i]));
33          neueMarkierungen.push(updatedSp[i]);
34          }
35          for(var i=0; i<neuImAbbSp.length; i++)
36          {
37          var ziele = FindQuellobjekte(FindTransformationsknoten(neuImAbbSp[i]));
38          neueMarkierungen.push(neuImAbbSp[i]);
39          }
40
41      //neue Umb-Transformation für alle Tasks, die noch
42      //nicht im Abbildungsmodell vorhanden sind
43      for(var i=0; i<neuImFp.length; i++)
44          {
45          newTransformationFromFt(neuImFp[i]);
46          }
47
48      for(var i=0; i<neuImSp.length; i++)
49          {
50          newTransformationFromSt(neuImSp[i]);
51          }
```

```
52      //Layout im Abbildungsmodell zurücksetzen
53      MakeLayout();
54      markiere(neueMarkierungen);
55       }
```

5.5.2 Auflösung der Konflikte im Abbildungsmodell

Für die Auflösung der Konsistenzkonflikte wurde in Kapitel 4 eine zweistufige Vorgehensweise vorgeschlagen. Der erste Schritt ist es, anhand der markierten Stellen im Abbildungsmodell im jeweiligen Prozess zu überprüfen und gegebenenfalls Anpassungen vorzunehmen. Eine weitere Anpassung ist bei Bedarf die Zuordnung der vorhandenen und neu erstellten Quell- und Zielobjekte zu den unterschiedlichen Transformationen. Dies sind die einzigen Anpassungen im Abbildungsmodell die manuell vorgenommen werden müssen. Das Hinzufügen und Löschen der Objekte geschieht automatisch.

Dazu muss im Abbildungsmodell markiert werden, welche Konfliktstellen überprüft wurden. Diese Markierung kann in ARIS im Abbildungsmodell nicht durch den Editor geschehen, weshalb diese Funktion wiederum in einem Report implementiert wurde, der ein entsprechendes Dialogfenster erstellt.

Dieser Dialog fragt für jede Transformation in der Konflikte aufgetreten sind, den Bearbeiter nach den zugehörigen Veränderungen in den Modellen.
Dieser kann entscheiden, ob die Änderung bereits vorgenommen wurde („bearbeitet“ -> grün markiert), oder ob dieses Element noch nicht bearbeitet wurde („zurückstellen“ -> gelb markiert). Die Task wird rot markiert, wenn sie durch den letzen Konsistenzcheck gelöscht werden darf, und besitzt damit auch die „bearbeitet“ Semantik.

Wenn sich alle Quell- und Zielobjekte im Zustand „bearbeitet“ befinden, kann durch einen letzten Schritt die Innere Konsistenz hergestellt werden. Der wichtige Quellcode des Reports ist für diesen Schritt wieder nachfolgend dokumentiert.

```
01 function EntscheideÄnderungen()
02     {
03     var abbildungsmodell = getAbbildungsmodell();
04     //Dialog zum Auflösen der Inkonsistenzen erstellen
05     var dlg = createDialog();
06     var result = Dialogs.show(dlg);
07
08     if(result == -1)
09         {
10         var a=0;
11         var tempResult = 0;
12
13         for(var i=0; i<neuImAbb.length;i++)
14             {
15         tempResult = dlg.getDlgValue(,,Objekt_‘‘+a);
```

```
16          if(tempResult == 0)
17              {
18              //Tasks nach der Auswahl entsprechend markieren
19              ...
20              a++;
21              }
22          for(var i=0; i<updatedFp.length;i++)
23              {
24          tempResult = dlg.getDlgValue(,,Objekt_‘‘+a);
25          if(tempResult == 0)
26              {
27          syncTimestamp(updatedFp[i], FindTaskImFachprozess(updatedFp[i]));
28              markiereGrün(updatedFp[i]);
29              }
30          else{
31              //behalte Markierung
32              }
33          a++;
34          }
35      ... Analog für die Sp-Tasks verfahren
36
37      //Layout im Abbildungsmodell zurücksetzen
38      MakeLayout();
39      }
```

5.5.3 Innere Konsistenz

Zur Herstellung der inneren Konsistenz betrachtet der letzte Konsistenzcheck zuerst die Beziehung der markierten Quell- und Zielobjekte und deren referenzierten Tasks. Angelehnt an Algorithmus 2 bei 4.3.5 beschrieben, wird dazu folgendermaßen verfahren:

- Ist die zugehörige Task vorhanden, wird das Quell- oder Zielobjekt entsprechend aktualisiert.
- Ist die zugehörige Task nicht vorhanden, wird das Quell- oder Zielobjekt im Abbildungsmodell gelöscht. Dies geschieht jedoch nur, wenn sich die Task im Zustand „bearbeitet“ befindet. Dies ist nochmals eine letzte Vorsichtsmaßnahme, um nicht durch eine vergessene Markierung eine Task ungewollt zu löschen.

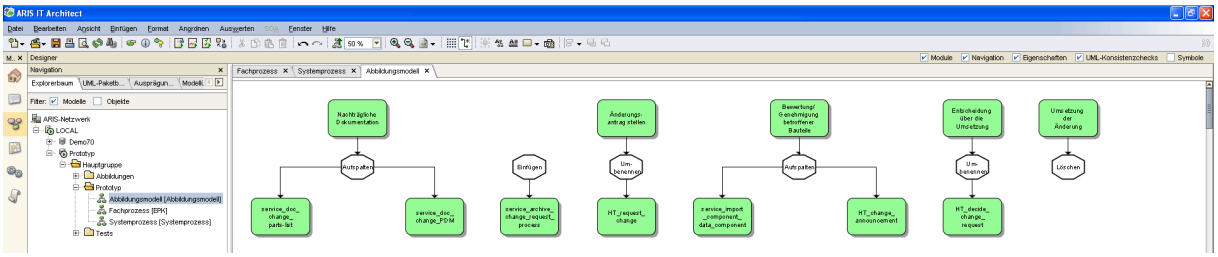

Abbildung 5.15: Das wieder hergestellte Abbildungsmodell

Somit werden automatisch alle Quell- und Zielobjekte entfernt, deren Beziehungen zu ihren Tasks nicht mehr existieren. Durch dieses Entfernen können die Transformationsknoten eine für ihren Typ ungültige Anzahl an eingehenden und ausgehenden Kanten zugewiesen bekommen. Daher wird in einem letzten Schritt der jeweilige Transformationstyp angepasst (siehe dazu 2.5.4).
Die nötigen Überprüfungen der eingehenden und ausgehenden Kanten der Transformationsknoten zur Herstellung der inneren Konsistenz sind mit relativ geringem Aufwand durchführbar. Der nachfolgende Quellcode beinhaltet diesen Schritt ebenfalls.

```
01 function InnereKonsistenz()
02     {
03     var abbildungsmodell = getAbbildungsmodell();
04         //jede rote Task entfernen
05     var abbTasks = abbildungsmodell.ObjOccListFilter(-1, Constants.ST_FUNC);
06     for(var i=0; i<abbTasks.length; i++)
07         {
08         if(abbTasks[i].getColor() == -52429)
09             {
10             abbTasks[i].Remove(true);
11             }
12         }
13         //jeden TK überprüfen, und richtig benennen
14     var transKnoten = abbildungsmodell.ObjOccListFilter(-1, 66389); //TK
15
16     for(var i=0; i<transKnoten.length; i++)
17         {
18         var quellen = FindQuellobjekte(transKnoten[i]);
19         var ziele = FindZielobjekte(transKnoten[i]);
20
21         if(quellen.length==0 && ziele.length==0){
22             transKnoten[i].Remove(true);
23             }
24         else if(quellen.length>1 && ziele.length>1){
25             split(quellen, ziele, transKnoten[i]);
26             }
27         else if(quellen.length>1 && ziele.length==0){           {
28             split(quellen, ziele, transKnoten[i]);
```

```
29              }
30          else if(quellen.length==0 && ziele.length>1){
31              split(quellen, ziele, transKnoten[i]);
32              }
33          else if(quellen.length==1 && ziele.length==0){
34              writeNodeName(transKnoten[i], ,,Löschen‘‘);
35              }
36          else if(quellen.length==0 && ziele.length==1){
37              writeNodeName(transKnoten[i], ,,Einfügen‘‘);
38              }
39          else if(quellen.length==1 && ziele.length==1){
40              writeNodeName(transKnoten[i], ,,Umbenennen‘‘);
41              }
42          else if(quellen.length==1 && ziele.length>1){
43              writeNodeName(transKnoten[i], ,,Aufspalten‘‘);
44              }
45          else if(quellen.length>1 && ziele.length==1){
46              writeNodeName(transKnoten[i], ,,Zusammenfassen‘‘);
47              }
48          }
49      //Layout im Abbildungsmodell zurücksetzen
50      MakeLayout();
51      }
```

In diesem Kapitel wurden die in ARIS Business Architect erstellten Reports vorgestellt, die einerseits die initiale Erstellung des Abbildungsmodells sowie andererseits die für diese Methodik nötige Erstellung (bzw. Änderung) des Systemprozesses vornehmen. Dadurch wurde die praktische Anwendbarkeit von Methodik 3 in einem Prozessmodellierungswerkzeug demonstriert.

Des Weiteren wurden die zunächst theoretisch untersuchten Konsistenzregeln auf diesen Prototypen angewendet, wodurch fachlich, oder technisch getriebene Änderungen im Abbildungsmodell als Inkonsistenzen erkannt werden können.

6
Diskussion und Ausblick

In diesem Kapitel werden verwandte Arbeiten, die Beziehungen zwischen der fachlichen Modellierung und der technischen Umsetzung herstellen, diskutiert. Außerdem werden offene Fragen für das Abbildungsmodell angesprochen, die in diesem Zusammenhang eine weiter reichende Verwendung des Modells betrachten.

Verschiedene Hersteller verfolgen Methodiken, um Software möglichst durchgängig zu modellieren. Diese Methoden sind jedoch meist an die entsprechenden Werkzeuge gekoppelt. Auf eine SOA bezogen, sollen einerseits fachliche Vorgänge erfasst und modelliert, und andererseits daraus bereits technische Modellelemente generiert werden. Sie beinhalten jedoch kein explizites Abbildungsmodell, sondern verringern nur in der Zusammenarbeit mit ihren Werkzeugen den Modellierungs- und Wartungsaufwand. Diese Ansätze basieren häufig auf der MDA (Model-Driven Architecture).

Die Firma MID führt mit ihrer „Open Modeling Platform - Innovator“ ein Werkzeug an, welches fachliche Prozesse erfasst, und eine Detaillierung bis zur technischen Ebene ermöglicht. In diesem Kontext teilt MID die Modellierungsebenen ähnlich dem Fachmodell-Systemmodell-ausführbaren Modell Ansatz dieser Diplomarbeit ein. Dazu werden Geschäftsprozesse anfangs frei modelliert, und Anwendungsfälle daraus abgeleitet. Diese führen zu einem plattformunabhängigen Analysemodell, welches die Grundlage für die technische Implementierung (dem Architektur- und Design-Modell) bildet. Dieses Analysemodell ist im Grundgedanken mit dem Systemmodell vergleichbar. Aus den dort spezifizierten Anforderungen an die technische Umsetzung wird anschließend das ausführbare Modell modelliert. Ebenfalls unterstützt durch den OMP - Innovator kommt dieser Ansatz mit einer eigenen Drei-Schichten-Architektur zum Einsatz [BEH+07].

Ähnliche Ansätze für eine plattformunabhängige Beschreibung der technischen Ebene mittels eines Zwischenmodells existieren auch von anderen Herstellern wie IBM und IDS Scheer. IBM schlägt für die durchgängige Modellierung beispielsweise eine eigene Referenzarchitektur für eine SOA vor, die solche Zwichenmodelle beinhaltet [AGA+08, AZE+07]. Der *Service-Oriented Solution Stack* (S3) beinhaltet neun Schichten, die eine SOA vollkommen spezifizieren kön-

nen. Für den Vergleich mit dieser Arbeit interessant, sind davon die *Service*-Schicht, und die Service-Component-Schicht. Auf unser Systemmodell bezogen, repräsentiert die Service-Schicht eher den oberen Teil, also die Assoziationen der Services zu den Fachtasks. Wohingegen die Service-Component-Schicht die bereits ausführbaren Elemente der Services identifiziert.
Die Spezifikation der SOA mittels dieses Referenzmodells ist jedoch weiterhin sehr technisch. Sie richtet sich von der Verständlichkeit eher an die technischen Mitarbeiter, da die vollständige plattformunabhängige Beschreibung der ausführbaren Services im Vordergrund steht. Für den Fachbereich wäre hier eher die „Lesbarkeit“ dieser Beschreibung interessant.

Eine andere Herangehensweise für eine durchgängige Modellierung sind Modelltransformationen, die ausgehend von dem fachlichen Modell ausführbaren Code (oft BPEL) ableiten [GX09, ODH, FKT08]. Dies geschieht oft durch das Erkennen und Anwenden von Patterns auf den fachlichen Prozess. Diese Patterns lassen einen Schluss auf die technische Realisierung zu und somit automatisch transformieren [All07, BFF$^+$08].

Wie S3 von IBM richtet sich der Ansatz von MID an die einfache initiale Erstellung einer SOA mittels durchgängiger Modellierung. Einerseits existieren Überlegungen, wie die praktische Vorgehensweise für die Implementierung einer SOA sein kann, jedoch bleiben weiterhin Fragen offen. Dazu zählen die Verständlichkeit durch den Fachbereich, die agile Anpassung an fachlich getriebene Änderungen und den flexible Reaktion auf technische Veränderungen.
Des Weiteren sind die beschriebenen Modelltransformationen nur ein Mittel um Durchgängigkeit einer SOA zu erreichen. Die in Kapitel 3 analysierten Vorgehensweisen zur Erstellung eines Systemprozesses haben gezeigt, dass die vollautomatische Transformationen oft nicht die Komplexität der fachlichen Anforderungen an den ausführbaren Prozess weiter geben, und einen manuellen Zwischenschritt weiterhin fordern.

Das Abbildungsmodell, welches in dieser Diplomarbeit genauer untersucht wurde, ermöglicht im Gegensatz zu den eben vorgestellten Methoden zusammen mit dem Systemprozess eine plattformunabhängige Beschreibung der Transformationen des fachlichen Prozesses in den ausführbaren Prozess. Außerdem kann es für die automatische Generierung des Systemprozesses verwendet werden, indem dort die Abbildungen der einzelnen Fachtasks auf die Systemtasks vor der Generierung bereits festgelegt werden. Des Weiteren bietet es im Nachhinein die Möglichkeit, diese Transformationen mit geringem Vorwissen nachzuvollziehen und ermöglicht somit eine schnellere Reaktion auf Änderungen in einer der Modellebenen.

Ausblick Das Abbildungsmodell wurde in dieser Arbeit ausschließlich auf die Abbildung von Elementen des Kontrollflusses untersucht. Datenobjekte werden aus Gründen der Verständlichkeit des Fachmodells häufig nicht auf dieser Ebene modelliert, jedoch kann dies in Einzelfällen notwendig werden. Die Modellierung mittels explizit modellierten Transformationen lässt eine Abbildung zwischen Datenobjekten jedoch ebenfalls zu, weshalb sich die Generierung des Abbildungsmodells mit Modifikationen ebenfalls auf diese Funktion anwenden lässt.

Ein weiterer Punkt betrifft die Verwendung des Abbildungsmodells (bzw. eines beliebigen Assoziationsmodells) für andere Anwendungsgebiete. Wenn in Fachmodell und Systemmodell beispielsweise das selbe Metamodell verwendet werden, ist eine technische Verfeinerung der Fachtasks in verschiedene Subprozesse durch das Abbildungsmodell denkbar. Somit wäre der Systemprozess eher eine technische Verfeinerung des Fachprozesses, als ein Zwischenmodell zwischen Fachbe-

reich und IT-Bereich. Dies kann auch für den Bereich zwischen Systemprozess und ausführbarem Prozess interessant werden.
Denkbar ist hier eine Abbildung der plattformunabhängigen Objekte des Systemprozesses auf die konkreten Implementierungsdetails der Zielplattform. Dies kann für die spätere Wiederverwendbarkeit und Portierung auf andere Zielplattformen genutzt werden.

Eine weitere mögliche Funktion betrifft die Art der Verwendung des Abbildungsmodells. Für die Erstellung Modells kann, entgegengesetzt zu dem hier verwendeten Top-Down Ansatz, auch Beschreibung von bestehenden ausführbaren Workflows zu Dokumentationszwecken im Vordergrund stehen. Im Zuge von bestehenden Legacy-Systemen, deren Funktionen teilweise nicht genutzt werden, können diese durch die „Insert“ Transformation im Abbildungsmodell erhalten bleiben. Diese sind somit in diesem Modell dokumentiert, und können bei einer späteren Erweiterung der Prozesse genutzt werden. Dies kann die redundante Implementierung bestehender Funktionalität in Ausnahmefällen vermeiden.

Während der Untersuchungen zu fachlich oder technisch getriebenen Änderungen der jeweiligen Prozess, wurden keine konkurrierenden Änderungen auf bestehende Transformationsknoten betrachtet. Diese können im ungünstigsten Fall zu einem Informationsverlust im Abbildungsmodell führen, da mehrere Bearbeiter schreibend darauf zugreifen. Eine Vorgehensweise an dieses Problem kann die Sperrung des Abbildungsmodells für Änderungen durch Dritte nach einem ersten Konsistenzcheck, mit gefundenen Inkonsistenzen, sein. Dies würde jedoch die Auflösung von Inkonsistenzen in den Modellebenen voraussetzen, bevor neue Konsistenzchecks durchlaufen werden dürfen. Hier können Methoden wie der gegenseitige Ausschluss durch Semaphoren und der Zeitpunkt der Anzeige der Inkonsistenzen weitere Themen für die Konsistenzerhaltung der Modellebenen werden.

Abschließend wäre ein aktiv durch eine Werkzeug unterstütztes Abbildungsmodell der nächste Schritt für weitere Untersuchungen auf die Akzeptanz des Abbildungsmodells. Die in diesem Prototyp verwendeten Scripte liefern noch nicht die gewünschte Benutzerfreundlichkeit, da mit der JavaScript Schnittstelle eine Funktion von ARIS genutzt wurde, die nicht vornehmlich für Modelltranssformationen und insb. die implementierten Änderungsoperationen ausgelegt ist. Somit wäre eine in ein Werkzeug integrierte Darstellung des Abbildungs- und Systemmodells empfehlenswert.

7 Zusammenfassung

In dieser Arbeit wurden vier verschiedene Methodiken vorgestellt, wie das Abbildungsmodell die Nachvollziehbarkeit im Systemmodell während der initialen Modellierung unterstützen kann. Dafür wurde der Fachprozess in den Systemprozess überführt, während das Abbildungsmodell die Umstrukturierungen des fachlichen Modells dokumentiert.
Dafür beschreiben jeweils zwei Methoden die Erstellung des Systemprozesses über das Abbildungsmodell und zwei Methoden die freie Modellierung des Systemprozesses. Nach der initialen Klärung, in welcher Reihenfolge die Modelle erstellt werden können, wurden automatisierbare Schritte identifiziert und die Voraussetzungen dafür beschrieben.

Das Systemmodell dient als Vorlage für die spätere Implementierung des ausführbaren Prozesses, wodurch die technische Detaillierung bereits auf dieser Ebene sehr formal geschehen muss. Die Beziehung zwischen fachlichen Tasks und technischen Tasks ist dadurch sehr schwer erkennbar, wodurch die Transformationen explizit im Abbildungsmodell dargestellt sollten.
Dies wurde erreicht, indem im Abbildungsmodell die sog. Transformationsknoten als Verzweigungselement zwischen Quellobjekten und Zielobjekten eingeführt wurde. Dieser gibt den Typ der Transformation an, und lässt eine formale Definition und somit automatische Überprüfung auf korrekte Umstrukturierungen zu. Die Quell- und Zielobjekte bilden für der Darstellung der Fachtasks und Systemtasks im Abbildungsmodell die jeweiligen Repräsentanten, und helfen nach erstmaliger Erstellung des Abbildungsmodells, Veränderungen in Fach- und Systemmodell zu bemerken.

Im Kapitel zur Untersuchung der Konsistenz wurden nach der initialen Erstellung des Abbildungsmodells plötzliche Änderungen an Fach- und Systemprozess betrachtet. Diese Prozessänderungen führen zu einer Differenz bzw. Inkonsistenz (in Bezug auf Attribute wie Timestamps) zwischen den Quell- und Zielobjekten des Abbildungsmodells und den Fach- und Systemtasks der Prozesse.
Das Abbildungsmodell wird dazu benutzt diese Inkonsistenzen aufzulösen. Dafür wurde eine dreistufige Methodik entwickelt. Diese besteht im ersten Schritt aus einer Anzeige und Markierung

der betroffenen Elemente. Diese markierten Problemstellen werden manuell in einem zweiten Schritt überprüft und die Prozesse dementsprechend angepasst. Im letzten Schritt stellt das Abbildungsmodell selbstständig in einem gesonderten Durchgang die Konsistenz wieder her, und kann dadurch auf nachfolgende Änderungen in den Prozessen wieder reagieren.

In dem Prototyp wurde eine der Methodiken zur Erstellung des Abbildungsmodells ausgewählt und in ARIS Business Architect umgesetzt. Bei der ausgewählten Methodik handelt es sich um Ansatz 3, welche ausgehend vom Fachprozess, diesen in das Metamodell des Systemprozesses transformiert, und das Abbildungsmodell anhand dieser beiden Prozesse generiert. Dazu wird das Abbildungsmodell zeitgleich mit dem Systemprozess durch einen Report erstellt, und mit einer initialen 1:1 Abbildung zwischen Fachtasks und Systemtasks gefüllt. Wie in Ansatz 3 beschrieben, werden in einem nächsten Schritt Anpassungen auf dem Systemprozess anhand von Änderungsoperationen vorgenommen. Diese geführten Veränderungen des Systemprozesses wurden wiederum durch Reports realisiert. Diese nehmen zeitgleich zu der Veränderung im Systemprozess Anpassungen im Abbildungsmodell vor.
Des Weiteren beinhaltet der Prototyp das Bemerken und die Reaktion auf Inkonsistenzen. Dafür wurde die dreistufige Vorgehensweise aus dem Konsistenzkapitel einzeln implementiert. Diese besteht aus dem Testen auf äußere Konsistenz und dem Markieren durch farbliche Hervorhebung der Tasks, dem manuellen Auflösen mittels eines Dialogfensters, und dem automatischen Herstellen der konsistenten Anzeige der Abbildungen durch die Transformationsknoten.

Literaturverzeichnis

[AGA+08] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Gariapathy, and K. Holley. SOMA: a method for developing service-oriented solutions. *IBM Syst. J.*, 47(3):377–396, 2008.

[All07] Thomas Allweyer. Erzeugung detaillierter und ausführbarer Geschäftsprozessmodelle durch Modell-zu-Modell-Transformationen. In Markus Nüttgens, Frank J. Rump, and Andreas Gadatsch, editors, *EPK*, volume 303 of *CEUR Workshop Proceedings*, pages 23–38. CEUR-WS.org, 2007.

[AZE+07] Ali Arsanjani, Liang-Jie Zhang, Michael Ellis, Abdul Allam, and Kishore Channabasavaiah. S3: A Service-Oriented Reference Architecture. *IT Professional*, 9(3):10–17, 2007.

[BBP09] Stephan Buchwald, Thomas Bauer, and Rüdiger Pryss. IT-Infrastrukturen für flexible, service-orientierte Anwendungen - ein Rahmenwerk zur Bewertung. In *BTW*, pages 526–543, 2009.

[BBR09] Thomas Bauer, Stephan Buchwald, and Manfred Reichert. Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen. 2009.

[BBR10] Thomas Bauer, Stephan Buchwald, and Manfred Reichert. Durchgängige Modellierung von Geschäftsprozessen in einer Service-orientierten Architektur. 2010.

[BEH+07] Alexander Bösl, Jan Ebell, Sebastian Herold, Christian Linsmeier, Detlef Peters, and Andreas Rausch. Modellbasierte Softwareentwicklung von Informationssystemen: Vom Geschäftsprozess zum Service-basierten Entwurf. *OBJEKTspektrum*, 04/07:46–54, jul 2007.

[BFF+08] Pascal Bauler, Fernand Feltz, Etienne Frogneux, Benjamin Renwart, and Céline Thomase. Usage of Model Driven Engineering in the context of Business Process Management. In Martin Bichler, Thomas Hess, Helmut Krcmar, Ulrike Lechner, Florian Matthes, Arnold Picot, Benjamin Speitkamp, and Petra Wolf, editors, *Multikonferenz Wirtschaftsinformatik*. GITO-Verlag, Berlin, 2008.

[BMW09] Jörg Becker, Christoph Mathas, and Axel Winkelmann. *Geschäftsprozessmanagement*. Springer, Berlin, Heidelberg, New York, 1 edition, 2009.

[BPM09] BPMN Specification 2.0 Beta v1. Technical report, Object Management Group (OMG), 2009.

[Erl07] Thomas Erl. *SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl)*. Prentice Hall PTR, Upper Saddle

River, NJ, USA, 2007.

[FKT08] Klaus-Peter Fähnrich, Stefan Kühne, and Maik Thränert, editors. *Model-Driven Integration Engineering. Modellierung, Validierung und Transformation zur Integration betrieblicher Anwendungssysteme*, volume Band XI of *Leipziger Beiträge zur Informatik*. Leipziger Informatik-Verbund (LIV), Leipzig, 2008.

[GX09] Shuai Gong and Jinhua Xiong. Interaction Mismatch Discovery Based Transformation from BPMN to BPEL. *Services Computing, IEEE International Conference on*, 0:292–299, 2009.

[Jos08] Nicolai Josuttis. *SOA in der Praxis: System-Design für verteilte Geschäftsprozesse*. dpunkt, Heidelberg, 2008.

[LVD08] Niels Lohmann, Eric Verbeek, and Remco Dijkman. Petri Net Transformations for Business Processes - A Survey. *Transactions on Petri Nets and Other Models of Concurrency*, 2008. (Accepted for publication).

[ODH] Chun Ouyang, Marlon Dumas, and Arthur H. M. Ter Hofstede. From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way.

[OMG05] Uml 2.0 superstructure specification. Technical report, Object Management Group (OMG), 2005.

[RD00] Manfred Reichert and Peter Dadam. Geschäftsprozessmodellierung und Workflow-Management - Konzepte, Systeme und deren Anwendung. *Industrie Management*, 16:23–27, 2000.

[Rei00] Manfred Reichert. *Dynamische Ablaufänderungen in Workflow-Management-Systemen*. PhD thesis, Universität Ulm, 2000.

[RvdA08] Nick Russell and Wil M. P. van der Aalst. Work Distribution and Resource Management in BPEL4People: Capabilities and Opportunities. In *CAiSE*, pages 94–108, 2008.

[Sch01] August-Wilhelm Scheer. *ARIS - Modellierungsmethoden, Metamodelle, Anwendungen*. Springer, Berlin [u.a.], 4. aufl edition, 2001.

[vdA98] Wil M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[vdAvDH+03] Wil M. P. van der Aalst, Boudewijn F. van Dongen, Joachim Herbst, Laura Maruster, Guido Schimm, and A. J. M. M. Weijters. Workflow mining: A survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267, 2003.

[vdAvH04] Wil M. P. van der Aalst and Kees van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, MA, USA, 2004.

[Wes07] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.

[YSWS08] JianHong Ye, ShiXin Sun, Lijie Wen, and Wen Song. Transformation of BPMN to YAWL. In *CSSE '08: Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, pages 354–359, Washington, DC,

USA, 2008. IEEE Computer Society.

[ZZ09] Liang-Jie Zhang and Jia Zhang. An Integrated Service Model Approach for Enabling SOA. *IT Professional*, 11:28–33, 2009.